Title: Radiation Detection Data Competition Report

Author(s): Anderson-Cook, Christine Michaela
Archer, Dan
Bandstra, Mark S.
Curtis, Joseph C.
Ghawaly, James M.
Joshi, Tenzing H.Y.
Myers, Kary Lynn
Nicholson, Andrew D.
Quiter, Brian J.

Intended for: Report

Issued: 2021-04-19 (rev.1)

Los Alamos National Laboratory
Technical Report LAUR-20-27292

# Radiation Detection Data Competition Report

Christine M. Anderson-Cook[1], Dan Archer[2], Mark S. Bandstra[3], Joseph C. Curtis[3], James M. Ghawaly Jr.[2], Tenzing H. Y. Joshi[3], Kary L. Myers[1], Andrew D. Nicholson[2], and Brian J. Quiter[3]

[1]Statistical Sciences Group, Los Alamos National Laboratory
[2]Radiation Detection and Imaging Group, Oak Ridge National Laboratory
[3]Nuclear Science Division, Lawrence Berkeley National Laboratory

September 2020

## Contents

## 1 Introduction

In FY2018 through FY2020, NA-22, the Defense Nuclear Nonproliferation Research and Development Program, funded a Data Science project to develop and implement statistical methodology to effectively host data competitions with the goal of leveraging the opportunity provided by crowdsourcing. By accessing and engaging expertise from a broader research community, there is an opportunity to attract innovative solutions from a variety of different research disciplines to advance the ability to solve important non-proliferation problems. This report summarizes the key results of this project after hosting two data competitions focused on urban radiation detection. The first competition was focused on attracting participants from the U.S. national laboratories, while the second, hosted by TopCoder, was open to the broader international community and awarded prize money to the top 10 competitors.

At the start of the project, there was strong interest from NA-22 to explore and develop the capability to host data competitions as a means of leveraging the broader community to solve important nuclear nonproliferation problems. Having a standard data set on which to compare different approaches based on clearly defined criteria was desirable to be able to evaluate the state of solutions for important problems. Initially, it was not clear that it would even be possible logistically and bureaucratically to host a competition with an international field of competitors and to award the prize money needed to attract solutions from top competitors. Happily, a path to host the competitions was ultimately found that allowed this powerful accelerator of improvements to be leveraged.

The motivation for hosting data competitions included:

1. The ability to attract innovative solutions from a diverse audience with varied backgrounds and expertise. It has been shown for centuries that encouraging broader participation can lead to accelerated improvement through ground-breaking innovative solutions that may have eluded the traditional subject matter experts [1]. In addition, many innovations come at the cross-section of different disciplines [2] and so the careful study of the competition solutions affords an opportunity to leverage these interdisciplinary advances and create a combined solution that leverages the best features of multiple individual solutions.

2. A fair competition with formal structure, carefully thought out objectives and a well-constructed data set provides a standardized means of comparison of different solutions. This allows a foundation for understanding the relative strengths and weaknesses of each, as well as the opportunity to understand the drivers of difficulty in the problem space.

3. The competitive fervor that a data competition inspires can drive rapidly accelerated improvements. By posting a leaderboard with real-time scoring, competitors are often more enthusiastic to work on improvements and the rate of advancement moves faster than in most other traditional development scenarios.

4. By exposing a broader community to interesting problems, new researchers including students with different types of expertise become aware of the focus of the data competition. This can be a possible recruitment tool or enable new interdisciplinary collaborations. For the urban radiological search competition, we were able to introduce this important class of problems to a large number of interested experts, with the hope for possible future engagement in this arena.

5. In addition, there was initially skepticism from those working in radiation detection that data science methods, such as machine learning, would be able to compete with domain knowledge expertise to adequately solve the problems of interest. The competition provided an opportunity to quantitatively compare different approaches to solving this well-established problem and gain understanding about the strengths and weaknesses of data-centric approaches and domain knowledge based methods.

The goal of the competition was to explore performance for detecting, identifying and locating six different radioactive sources in the simulated urban environment provided by the MUSE data generation capability. This scope was determined by a team of experts at ORNL based on the ability to generate suitable data, the recommended upper bound for the size of the data set provided to the competitors, and the ability to model the results with sufficient sample sizes to evaluate and test each of the questions of interest.

Data were generated across a wide variety of different scenarios, and the inputs for these scenarios were chosen to mimic the breadth of urban environments seen in practice. The choice of which inputs to manipulate, and which ranges to consider were based on factors known to impact performance of radiation search algorithms and what conditions were likely to be encountered in U.S. urban environments.

The data were generated using MAVRIC[3] a stochastic simulation code developed at Oak Ridge National Laboratory distributed as part of the SCALE package[4]. The input factors varied across the competition data were divided into several categories: radiological background characteristics, radiological source types and configurations, and detector movement. To create a diverse radiological background, multiple versions of urban streets were used with different configurations and compositions for the buildings and features. Five different radioactive source types were included, and shielding around the sources was varied. A sixth source comprised of the combination of two of the sources ($^{99m}$Tc + HEU) was also included. These sources range from weapons grade materials to isotopes common in medical or industrial applications:

1. HEU: Highly enriched uranium
2. WGPu: Weapons grade plutonium
3. $^{131}$I: Iodine, a medical isotope
4. $^{60}$Co: Cobalt, an industrial isotope
5. $^{99m}$Tc: Technetium, a medical isotope
6. A combination of HEU and $^{99m}$Tc

In addition, some characteristics of the detector's movement were varied. The speed of the detector moving along the urban street was fixed throughout a run, but was varied between runs across the range of anticipated operational use. The speeds considered ranged from slow walking speed to moderate driving speed.

To keep downloads and manipulation of the data manageable for the competitors, the target total file size for the zipped data was 10 GB (a recommended ceiling by several data competition websites). The data set consisted of approximately 10000 "runs" in the training set, and approximately 16000 runs in the test set (with 42% in the public test set, and 58% in the private set). The public test data are used to provide a real time score during the competition. The private test data are used to determine the final ranking of the algorithms, but no feedback on performance on this subset of data is provided until after the competition has concluded.

For each run in the test set, the competitor specifies (a) whether a source is present (many of the runs did not contain any source), (b) which of the sources it is and (c) at what time during the run was the detector closest to the source. Additional details about the competition are available [5] [6].

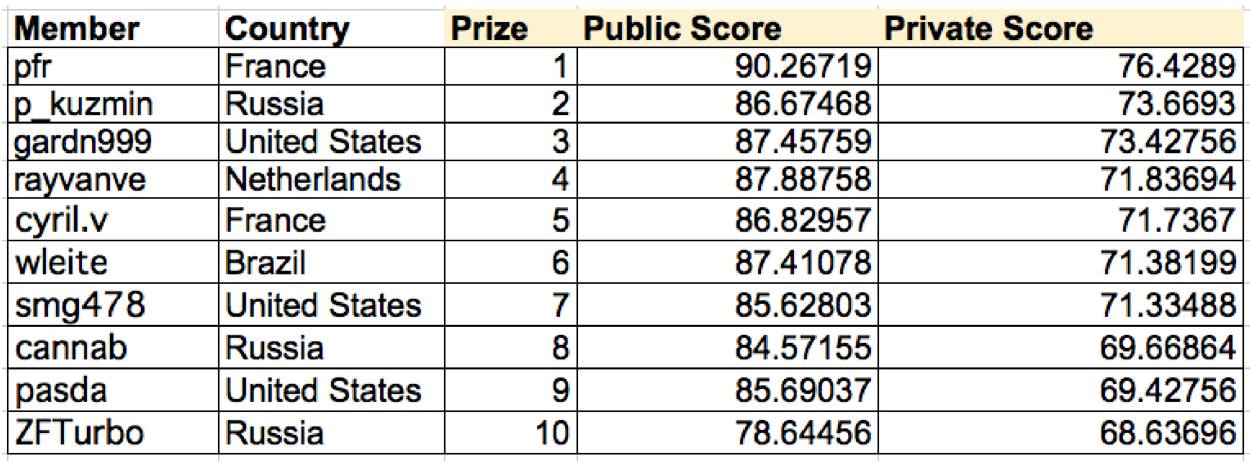
The first competition that focused on government competitors, `https://datacompetitions.lbl.gov/competition/1/`, ran from January to May of 2018 and attracted 16 teams, comprising 25 federally funded researchers who contributed a total of 1016 submissions throughout the duration of the competition.

The second competition, `https://www.topcoder.com/challenges/30085346`, was open for any participants and was hosted by a professional data competition provider, TopCoder. It ran from March to April 2019 and attracted 1614 submissions from 71 competitors. The participants in this second competition came from many different countries and had diverse technical backgrounds.

Figure 1 shows the results of the TopCoder competition. The final prizes were distributed based on the competitors' scores on the private portion of the test set. Scores based on the private subset of the test data were available in real time to the competitors while the competition was running. The TopCoder platform facilitated access to a diverse, experienced and international set of competitors.

## 1.1 Problem Space

The goal of the competition was to focus on algorithms that received data from a specific detector moving through an urban environment. A successful algorithm is able to (1) detect when an anomaly from the background has occurred, (2) identify which of several alternative source types it is, and (3) locate the time (or location) when the detector was closest to the detected source.



| Member | Country | Prize | Public Score | Private Score |
|---|---|---|---|---|
| pfr | France | 1 | 90.26719 | 76.4289 |
| p_kuzmin | Russia | 2 | 86.67468 | 73.6693 |
| gardn999 | United States | 3 | 87.45759 | 73.42756 |
| rayvanve | Netherlands | 4 | 87.88758 | 71.83694 |
| cyril.v | France | 5 | 86.82957 | 71.7367 |
| wleite | Brazil | 6 | 87.41078 | 71.38199 |
| smg478 | United States | 7 | 85.62803 | 71.33488 |
| cannab | Russia | 8 | 84.57155 | 69.66864 |
| pasda | United States | 9 | 85.69037 | 69.42756 |
| ZFTurbo | Russia | 10 | 78.64456 | 68.63696 |

Figure 1: Leaderboard for the TopCoder competition with the public and private scores for the top 10 prizewinners.

More formally, nuclear search operators are tasked with finding faint signatures of illicit nuclear sources in strongly varying background environments Radiological backgrounds a most typically due to the radioactive decay of naturally occurring radiological material (NORM), of which potassium-40 ($^{40}$K), uranium (U) and its radioactive daughters, and thorium (Th) and its daughters, or KUT. In an urban search scenario, operators depend on real-time algorithms running on their detector units to alert them to the presence of radioactive sources while driving or walking through city streets. When detected, the operators respond to find and verify the origin of the source. False positives, an alert that there is a radioactive source present when it is not, cost time and response effort. For algorithms to be successfully deployed, the number of false positives must be bounded to reduce the chance of alarm-fatigue. Too many false positives could cause operators to lose confidence in the algorithm and begin ignore alerts at important times. Alternately, false negatives, when no alarm registers when threat sources are actually present, could have disastrous consequences if dangerous situations go unaddressed. Therefore the desired solutions to the urban radiation search problem require an appropriate balance between false positives and false negatives. It is also important when comparing alternative solutions to be able to understanding algorithm sensitivity and false positive rates when designing a search operation.

The data competitions provide a formal mechanism to test radiological search algorithms against a set of synthetic data, enabling the development, comparison and evaluation the algorithms and their approaches.

The data set contains representative time series detector response functions of a 2 in.×4 in.×16 in. NaI(Tl) detector moving through a city street. Gamma-ray detectors, specifically NaI(Tl) (thalium-doped sodium iodide) detectors, are commonly used for urban radiation surveillance because of their low cost and high gamma-ray detection efficiency. Many illicit sources of interest, including industrial sources and special nuclear material, emit gamma rays as they undergo radioactive decay. Gamma rays are able to travel long distances and penetrate through dense material, making them ideal signatures for detection during search operations. Radioactive isotopes can be identified by the characteristic gamma rays they emit. Because shielding, or dense material around the source, attenuates gamma rays and can dramatically change the spectral shape of the gamma-ray flux, radiation source search algorithms must be robust to many shielding configurations to correctly detect and correctly identify illicit sources. In addition, discerning between gamma rays emitted by benign isotopes (such as medical or industrial isotopes) and illicit sources (such as special nuclear material) is important to guide the appropriate response and to prevent unnecessary overreactions. Hence detection and identification based on data obtained from a NaI(Tl) detector is a natural focus for a data competition.

To generate a realistic background against which sources can be introduced, the stochastic simulation code focused on generating data representative of the KUT present in everyday materials such as brick, concrete, and asphalt. KUT activities are highly variable in an urban search environment due to the fact that nearby buildings can emit dramatically different background signatures. The environment considered for the data competition was simplified to not include clutter, such as people, cars or other mobile objects. Only key stationary objects on the street were incorporated.

To spur innovation in radiation detection algorithms, a large data set, based on Monte Carlo radiation transport simulations, with realistic gamma-ray backgrounds and simulated sources was developed. A key advantage of using the MUSE stochastic simulation model is that the backgrounds are generated using Monte Carlo radiation transport simulations, meaning the background composition and variability can be easily modified. Having a diverse set of background that is representative of a variety of locations, regions, layouts and the resulting variability is important for the evaluation of different algorithms' performance, considering the known difficulty in detecting weak source signals relative to urban background. This approach different from other efforts to generate radiological source terms which used experimental data, Monte Carlo simulations [7], detector response simulation codes (like GADRAS) [8] and/or solid angle calculations [9]. Source terms are usually injected into a measured background or a simple model [10].

A frequent problem with observed data sets is that it is difficult to know what ground truth is. Hence a significant advantage of using simulated data is that everything in the scene is known. As a result, high-quality labels can be applied to each synthetic data set to train and evaluate algorithms. Moreover, the benefit of a unified data set for comparison enables multiple algorithms to be evaluated on a common data set to understand and characterize different algorithms' performance in a variety of conditions such as detector speed, background variability, source type, and intensity.

By combining flexible data generation code with newly developed statistical design methods, we were able to generate a structured set of data to be presented to the competitors [5]. The post-competition analysis methods provide opportunities for both exploratory data analysis and model-based methods to formalize relationships between different environmental and source configurations [6].

### 1.2 Desirable Characteristics of the Competition Data Set

In generating the data to be presented to the competitors, it was important that the data had the right characteristics to drive innovation to improve objectives of maximum importance. It was also important that the data had sufficient quality and volume to be able to interpret the results and determine how the different algorithms compared for the various objectives of detect, identify and locate. The key characteristics that were sought in the data set included the following features:

**Data set size** Adequate size to be able to evaluate all of the effects from different inputs of interest, but of manageable size to not deter participation by many competitors with diverse backgrounds.

**Format of data** The structure of the data needed to match what experts in radiation detection are familiar with, but also allow for an easy learning curve for participants who are not already familiar with this area. We took care to provide supporting background materials to competitors to make it straightforward for those new to urban radiation detection problems to learn sufficient background to participate, as this will encourage a diverse competitor pool.

**Space-Filling** The data sets should provide coverage throughout the input space of interest. The input space is defined as the set of combinations of all of the inputs of interest that the competition seeks to explore and characterize during the competition. The goal is to have the data set fill the ranges of each of the inputs adequately to support model estimation for all input factors. Since the competition explored multiple sources, it is possible to think about the complete test data set as a set of 7 separate data sets - one for each source, plus the collection of no source runs. An ideal overall test data set has adequate space filling for each of these data sets. A final step before the data are presented to the competitors is to randomize the order of the runs in the overall data set.

**Realism** To the extent possible from the simulator, the data should be presented in a way that matches how solutions will be deployed in practice. The nature of competitions with the full data set being available at the start of the competition, and the ability to resubmit solutions to receive feedback on performance are both constraints on being able to fully realize this objective. In addition, we recognized early that it would not be possible to adequately explore all aspects of radiation detection in a single competition, so intentional choices were made to bound the problem. For example, omitting clutter and detector variability helped to limit the dimensionality of the input space.

**Level of Difficulty** The data sets (training, public test and private test) should target regions that will enable differentiation between competitor performance. this suggests that the ideal data set would avoid too much data in regions where everyone is expected to get the answer correct, or everyone is expected to get it wrong. In addition, new methodology was developed to allow creation of the three data sets to show a progression from an emphasis on the easiest versions of the problem (training) to hardest (private test), as this enables the host to assess the ability of algorithms to adapt to new more challenging scenarios. This approach allows for the host to gain information about the algorithms' ability to solve new scenarios that represent an extrapolation to new cases in the input space.

**Avoid Exploitable Artifacts** Since the competitors' primary objective it to win prize money, it is important to present the data in a way that avoids artifacts that can be exploited by the competitors to answer the question of interest in a way that does not translate into an operational solution.

**Structure of Leaderboard** The leaderboard scoring metric will drive improvements in different areas by rewarding certain characteristics in the solutions. It is critical that the leaderboard scoring appropriately penalizes and rewards characteristics in a way that is proportionate to how much they are valued in the operational solution. For our competition, it was important to find the right balance between penalizing the false positive and false negative rates.

## 1.3 Report Organization

This report provides a detailed summary of key outcomes and learning from the "Developing and Analyzing Competitions for OPD" project. Section 2 provides a summary of the current state of radiation detection in an urban environment. Section 3 gives an overview of the radiation transport simulations used to generate the data used in the competition. Section 4 contains an overview of how the format of the data for each of the runs in the competition was created, how the input space of interest was explored with different emphases for the training, public and private test data sets, and the initial testing of data and the competition concept through the government-only competition. Section 5 provides details about the steps required for hosting a data competition, including the development of competition goals, constructing data sets to match the chosen goals and constructing a realistic leaderboard scoring mechanism to reflect the relative priorities of multiple competition objectives. It also summarizes interactions with an established hosting web company, running the competition, and tools available in a post-competition analysis to extract maximal information and understanding from the results. Section 6 provides results from the open international TopCoder competition, with numerical and graphical summaries for different aspects of the top competitors' algorithms. The section also includes comparisons between the winning algorithms by considering many aspects of the detect, identify and locate objectives for each of the 6 sources considered. In addition, some of the highlights of the new statistical methods developed as part of the project are described. Section 7 provides details about the approaches and inner workings of the top algorithms. Some of the methods that were used are discussed, as well as an assessment of how well the methods might be adapted to operational scenarios. Section 8 provides some final conclusions, lessons learned from the competition, and discussion about future research and development recommendations.

# 2 Current State of Radiation Detection

Here we will review the landscape of algorithms and approaches that are relevant to the gamma-ray detection problem at issue in the data competition, that is, the problem of urban search. We will begin with background information about gamma-ray physics and detectors (Section 2.1), then perform a survey of different methods (Section 2.2), and finally conclude with a summary (Section 2.3).

## 2.1 Gamma-ray physics

The physics of how a gamma ray goes from impinging on a detector to being recorded as an event by the data acquisition system is complex, yet fundamental to understanding the signals that any detection algorithm needs to grapple with. Here we will review the physics of gamma-ray interactions and the physical processes used to convert their signatures into electronic data when using different kinds of detector materials.

### 2.1.1 Gamma-ray Interactions with Matter

Gamma rays primarily interact with matter via three processes: 1) photoelectric effect, 2) Compton scattering, and 3) pair production [11, 12]. See Figure 2 (a)-(c) for cartoons of each. For photoelectric and Compton processes, gamma rays typically interact with electrons bound in atomic matter in the environment. In pair production, the photon annihilates in the electric field (typically near an atomic nucleus) to create an electron-positron pair.

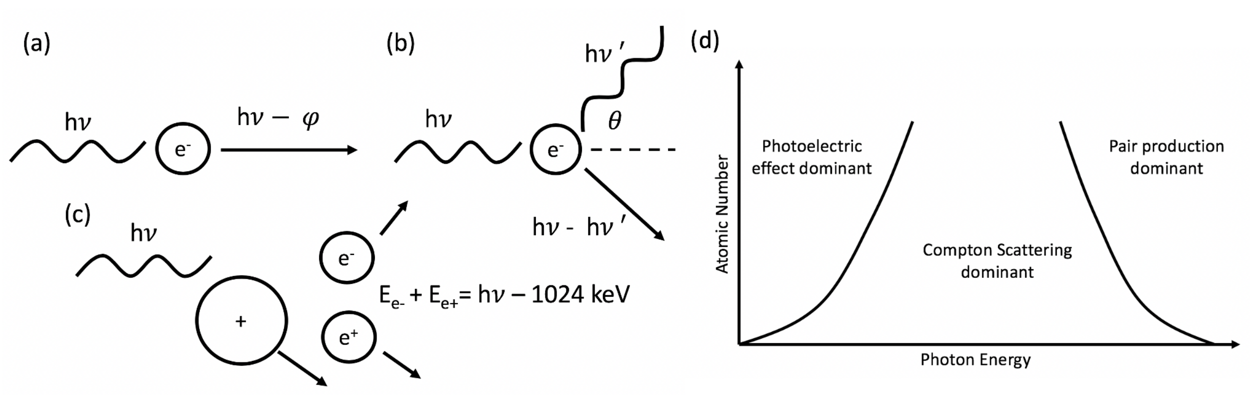


Figure 2: Cartoons of (a) photoelectric interaction, all of the gamma-ray energy is transferred to an electron, (b) Compton scattering, where only part of the gamma-ray energy is transferred and (c) pair production, the gamma ray interacts in the electric field to create an electron-positron pair and transfers some momentum to the nucleus. (d) Cartoon of gamma-ray interaction probability with matter as a function of photon energy and atomic number.

In photoelectric interactions, all of the photon energy, $h\nu$, is transferred a bound atomic electron. The result is that the parent atom is ionized, and the vacancy in the electronic orbit of the atom is quickly filled by electrons in higher-energy orbitals, releasing X-rays in the process. The final energy of the ionized electron is $h\nu$ - $\phi$, where $\phi$ is the binding energy of the atomic orbital. Photoelectric interactions are dominant at lower energies (typically below about 1 MeV for NaI(Tl)) but quickly become less probable at higher energies [13]. See Figure 2d for a schematic describing the most probable gamma-ray interaction type as a function of atomic number and energy. Photoelectric interactions are directly useful for gamma-ray spectroscopy, producing photoelectric peaks or "photopeaks" in gamma-ray spectra, which can be used to identify the parent radionuclide.

In Compton scattering, only part of the gamma-ray energy is transferred to the electron, resulting in a lower energy gamma ray and an excited electron. The cross section for this interaction is dominant in NaI(Tl) between about 1 and 7 MeV. Because only part of the gamma-ray energy is imparted, this interaction is less useful for gamma-ray spectroscopy, but in larger detector volumes (such as 2x4x16" NaI(Tl)) it is not uncommon for the Compton scattered gamma ray to undergo multiple interactions within the detector, resulting in full energy depositions that are difficult to distinguish from photoelectric events. Compton scattering can be used to infer position information on the source usually by determining the angle of photon deflection between two successive interactions, but the necessary information to enable this type of inference was not provided to competitors in this project. However, in gamma-ray spectroscopy, Compton scattering can produce noise or the "Compton continuum" which masks photopeaks in spectra.

Pair production only becomes possible at photon energies greater than 1022 keV (the sum of the electron/positron rest masses) and becomes the dominant interaction in NaI(Tl) at 7 MeV and beyond. Once the positron slows down, it will annihilate with an electron producing two 511 keV gamma rays. If this process happens in material outside of the detector, these 511 keV gamma rays can be detected producing a photopeak in the detector. If pair production occurs inside of a gamma-ray detector, the energy deposited by the electron-positron pair will also be detected and the 511 keV photons produced by electron-positron annihilation may also be detected via photoelectric interaction or Compton scattering. If both 511 keV photons escape, the result is a peak at $h\nu$ - 1022 keV. If one 511 keV photon interacts via the photoelectric interaction the result is a peak at $h\nu$ - 511 keV. If either of the 511 keV photons interact via Compton scattering, they are added to the Compton continuum producing no photopeaks. In gamma-ray spectroscopy, pair production is common when high energy gamma ray are produced after neutron capture or inelastic neutron scattering with matter. The presence of a 511 keV photopeak and photopeaks at $h\nu$ - 1022 keV and $h\nu$ - 511 keV are typically signs of neutron sources.

### 2.1.2 Gamma-ray Detectors

Gamma-ray detectors used for non-proliferation activities generally fall into three categories: 1) organic scintillators, 2) inorganic scintillators and 3) semiconductor detectors. Scintillator detectors produce many photons per gamma-ray interaction in the visible to ultraviolet bands. These photons are collected and converted to a voltage pulse by some kind of photo-sensor coupled to electron multiplication electronics (typically by photomultiplier tubes or more recently silicon photomultiplers), converting the light into an electronic signal. The integral of this signal is proportional to the number of low energy scintillation photons, which is also proportional the the energy deposited by the gamma-ray interaction. In semiconductor detectors, ionization by secondary electrons create many electron-hole pairs in the material. The number of electron-hole pairs is proportional to the incident gamma-ray energy. The net charge collected is therefore proportional to the energy deposited by the gamma ray.

Organic scintillators are composed of organic molecules, typically in the form of a crystal (e.g., stilbene), liquid (e.g., Eljen Technologies EJ-301), or a plastic (e.g., polyvinyltoluene) [11, 12]. Plastic and liquid organic scintillators can be manufactured to a wide variety of geometries and can be made into fairly large detectors, while crystal scintillators tend to be smaller. In organic scintillators, gamma rays ionize electrons in organic molecules causing florescence as the molecule relaxes to its ground state. The time profile is dependent on the incident ionizing particle type, so pulse shape discrimination can be used to detect different kinds of radiation in a single detector. Unfortunately, organic scintillators are not useful to gamma-ray spectroscopy because they have low atomic numbers, meaning that Compton-scattering interactions dominate (see Figure 2 (d)). Therefore gamma rays tend to leave the scintillator without undergoing photoelectric absorption so photopeaks are typically not present in the spectrum.

Inorganic scintillators are typically single crystals which may include dopants to modify electron energy band structure and to introduce trapping sites for ionized electrons. Scintillation in these crystals is tied to the electron energy band structure of the entire crystalline structure, instead of exciting molecules as in organic scintillators. When a gamma-ray interacts with an electron and it becomes excited, as it relaxes it produces scintillated light, usually in the visible to ultraviolet regime. Typical examples of inorganic scintillators include NaI(Tl), CsI, LaBr and $SrI_2$. Photoelectric interactions are favored, at least up to 1 MeV in these materials, so they are good candidates for gamma-ray spectroscopy. The energy resolution, or width of the photopeak, varies from material to material, and typically most inorganic scintillators have low to moderate energy resolution. Low to moderate energy resolution makes resolving photopeaks that are close together in energy difficult. The maximum size of inorganic scintillator crystals varies widely between materials, where NaI(Tl) can be reliably made in 2-in×4-in×16-in geometries, but LaBr has a maximum size of 2-in diameter by 2-in height cylinders.

Semiconductor detectors are typically single crystals of semiconducting material, meaning there is a well defined band gap between conduction and valance bands. This band gap needs to be wide enough to minimize the probability that thermal fluctuations elevate valance band electrons into the conduction band, a potential source of noise. If the band gap is not sufficiently wide, these materials need to be cooled, to cryogenic temperatures in the case of high purity germanium (HPGe) or room temperature or below in the case of CdZnTe (CZT). When gamma rays interact with electrons in the semiconductor they are excited from the valance band to the conduction band and are collected directly via an applied electric field. Typically semiconductor detectors have moderate to high energy resolution, because ionized electrons are detected directly without the need for scintillation. However, these devices are usually small (about 1 cm$^3$ for CdTe detectors) so their detection efficiency is low. In addition, some have to be cooled to cryogenic temperatures making them more expensive and harder to carry.

## 2.2 Gamma-ray Detection and Identification Algorithms

Gamma-ray detection and identification algorithms seek to take the photon events recorded by a gamma-ray detector and convert them into information about whether the current radiation environment is somehow anomalous when compared with background conditions, and if so what isotope(s) is causing the anomaly. The particular detection problem posed by the data competition, that of detecting and identifying a lone (or “orphan”) radioactive source using ground-based mobile gamma-ray detectors, has been studied for many years (e.g., [14, 15, 16]). The fundamental difficulty of the detection and identification problem arises from the highly variable backgrounds often encountered by mobile systems, the variety of potential spectral shapes of anomalous sources, and the limitations of counting statistics [16]. We will restrict this discussion to algorithms that take in only photon event time and energy data, such as was provided to the competitors in the competition, and not position or other contextual data. We will also restrict the discussion of identification to the context where the anomalous isotope is present at low signal-to-noise (SNR), instead of the general problem of identifying the sources of many gamma-rays signatures in spectra, although such studies are related (Section 2.2.5 will briefly touch on these methods).

The problem of gamma-ray anomaly detection is made difficult by the complexity of the natural gamma-ray background, primarily due to the presence of $^{40}$K, the $^{238}$U series, and the $^{232}$Th series (the KUT backgrounds), which are present in soil and building materials and whose concentrations may vary by orders of magnitude [17]. Other background contributions are from $^{222}$Rn progeny in the atmosphere ($^{222}$Rn and its progeny are part of the $^{238}$U decay series) and line and continuum emission from cosmic ray interactions in the atmosphere [18]. In addition to the complexity of the background, the statistical nature of the photon counting problem adds a level of statistical variability to the problem, which is difficult to disentangle from the true nature of the shifting background and the presence of any potential nefarious sources. In addition, because of Compton scattering both in the environment and in the detector, the anomaly isotopes are rarely pure photopeaks but contain various amounts of downscatter, making it possible that a given isotopic anomaly may present itself differently under different physical conditions.

A common feature of detection and identification algorithms is that they must be given some sample of background data, from which the algorithm will derive a set of parameters to describe the background. For identification, training spectra that include the specific anomalies to be identified are also likely to be required so that the algorithm can learn the parameters needed to maximize its sensitivity and specificity for those particular isotopes. This process of tuning an algorithm with background and source data is typically called *training*.

Gamma-ray detection algorithms take many forms, depending on a variety of factors and requirements for the specific application. A non-exhaustive list of the qualities of different algorithms is the following:

- Whether it operates primarily in the time domain, primarily in the energy domain, or in both domains;
- Whether it performs only anomaly detection, only anomaly identification, whether it simultaneously performs both tasks, or whether it performs one prior to the other;
- Whether it uses the full spectrum or a portion of the spectrum;

- How the background is estimated: e.g., is it fixed to an average, predicted using a moving average, or predicted using nearby spectral regions; and
- How/whether context other than time is used.

Because of the statistical nature of photon counting, most detection algorithms amount to a statistical test: a particular test statistic is chosen, the distribution of the statistic under background conditions is estimated through training, and the relevant statistical test is performed (e.g., a $\chi^2$ or likelihood ratio test). Algorithms are distinguished by their choice of statistic, by their methods for estimating the statistic's background (or null) distribution, and by their methods for maximizing the signal that particular anomalies produce in that statistic. In what follows, we will review several detection and identification algorithms, emphasizing their key design choices, their strengths, and their limitations.

#### 2.2.1 A prototype algorithm for anomaly detection: Gross counts k-sigma

The simplest gamma-ray detection algorithm is gross counts $k$-sigma (e.g., [19, 20]), and it can be used to illustrate the basic features of more complex algorithms.

For this algorithm, the measured spectrum is summed across all energies to obtain the gross counts, say $N_i$ for measurement $i$. Assuming constant integration time $\Delta t$ for simplicity, the average gross counts per measurement of background ($\bar{N}$) is calculated from a set of background data. The main assumption of the method is that the gross counts $N_i$ follow a Gaussian distribution of mean $\bar{N}$ and some variance var$[N]$; if the distribution is truly governed only by random fluctuations it can be considered Poissonian, then var$[N] = \bar{N}$, although other choices could be made, such as the variance of recent measurements. The test statistic for gross counts $k$-sigma is the number of standard deviations away from the mean:

$$z_i \equiv \frac{N_i - \bar{N}}{\sqrt{\mathrm{var}[N]}}, \tag{1}$$

meaning the $z_i$'s null distribution is a unit normal. When $|z_i|$ exceeds a chosen threshold $k$, the measurement is deemed anomalous.

Using the null distribution, the threshold $k$ can be set by choosing a desired False Alarm Rate (FAR). From the FAR and integration time $\Delta t$, $k$ can be calculated by integrating the tails of the unit normal distribution. For example, a FAR of 1 per 8 h ($3.47 \times 10^{-5}$ s$^{-1}$) and an integration time of 1 s leads to a $k$-sigma threshold of 4.14, so anomalies would need to have approximately "4-sigma" significance in the common usage.

Unfortunately, these simple assumptions are often violated in situations encountered by mobile spectrometers because the gross counts variability is often governed by shifts in the true underlying distribution, and therefore are non-Poisson (e.g., Figure 3), so various modifications must be made in the time and energy domains in order to make it useful.

One modification in the time domain is to use a recent average of the measured gross counts for $\bar{N}$ and the associated variance. There are also examples of eschewing the Gaussian distribution altogether and instead using the empirical distribution of past measurements and comparing it to the distribution of the latest few measurements using Kolmogorov-Smirnov (KS) or related tests [14, 21].

A modification in the energy domain is to use the gross counts in only a region of the spectrum, not the entire spectrum. This modification introduces some specificity to desired anomalies, e.g., the region containing the $^{137}$Cs photopeak could be chosen, however the assumption of large mean counts ($\gtrsim 30 - 50$) must be maintained if the Gaussian assumption is used, although extension of the test statistic to the Poisson regime is straightforward. In addition, variability in background isotopes at higher energies can "bleed down" into the chosen window due to incomplete energy deposition of the background lines (primarily due to Compton scatter), which will lead to systematic changes in the distribution of counts in that window. For example, detection of anomalies in a $^{137}$Cs photopeak window is greatly improved once one accounts for the downscatter of $^{40}$K into the window [22].

#### 2.2.2 Time-domain anomaly algorithms

Some algorithms for detecting anomalies operate in the time domain only, with little spectral information used. These algorithms take advantage of the inherently different time evolution of anomalous sources,

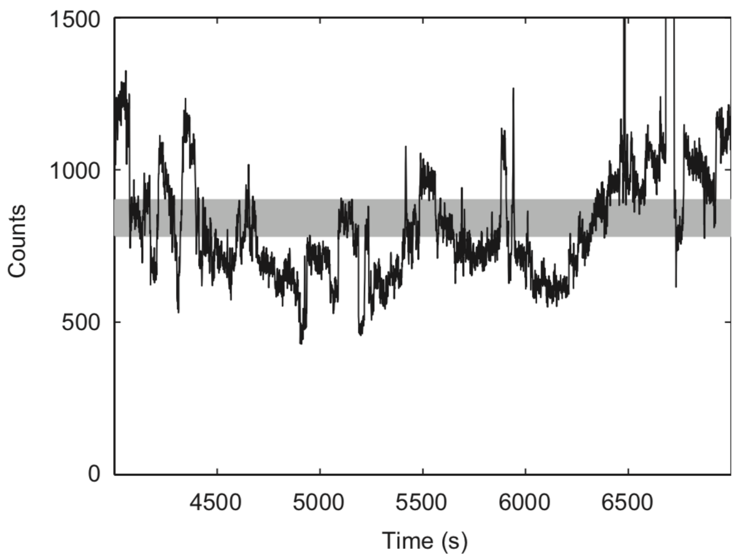


Figure 3: Figure 1 from [19] showing a measured gross count rate and a $\pm 2\sigma$ band around the mean gross count rate for a mobile detector system. The gross count rate in urban search problems typically does not follow a fixed Gaussian distribution.

whose fluxes generally follow the inverse square law due to their point-like nature, and the time evolution of the background, which generally arises from distributed areas like buildings and the soil and thus presents differently from a point source. These algorithms come mainly from the radiation portal monitor (RPM) literature, which has the benefit of being able to assume relatively constant background rates, but they are worth examining as a class since the data competition scored approaches based on their ability to identify the time when an anomaly occurred.

**Matched filters in the time domain** In [23], the authors compare RPM count rate data to the idealized profile of a point source passing by a planar detector at a known speed and standoff distance. The idealized profile is used as a filter, and it is convolved with the count rate data in order to find any times of high correlation, indicating that a source may be present (e.g., Figure 4).

This method has also been applied to the case of a mobile detector system [24], where the source is assumed to be stationary but the detector is moving at a known speed. Since the distance to the source is unknown, unlike in the RPM case, different matched filters are convolved for different source standoffs and the highest signal is used to determine both the time of closest approach and the most likely standoff.

In more recent work, maximum likelihood was used to find the best-fit point source model in an RPM context [25]. By fully modeling the Poisson statistics of the problem and utilizing maximum likelihood theory to determine detection thresholds, they found improved detection capability for low SNR scenarios.

These approaches are sensitive to background variability, which will increase the probability of false alarms. Often the count rates used are not gross counts but counts in a spectral window of an isotope of interest (e.g., $^{137}$Cs), which may reduce some background variability and gives the approach some limited ability to perform identification.

**Imaging methods** Some time-domain algorithms exploit time-dependent point-source signals induced by collimation. These systems require multiple detectors arranged in a known configuration relative to the collimator(s). By using even a simple imaging configuration, the time variability of background can be reduced by such methods because the time signal of a point source presents differently in each detector, while changes in background appear approximately identical in each detector due to their diffuse (non-point) origin — thus part of the time filter is encoded in hardware, and some of the time filter must be done in software (to determine the unknown source standoff). Examples of such systems and algorithms are a mobile system that used a one-dimensional coded mask [26, 27], and even a simple system where a simple piece of

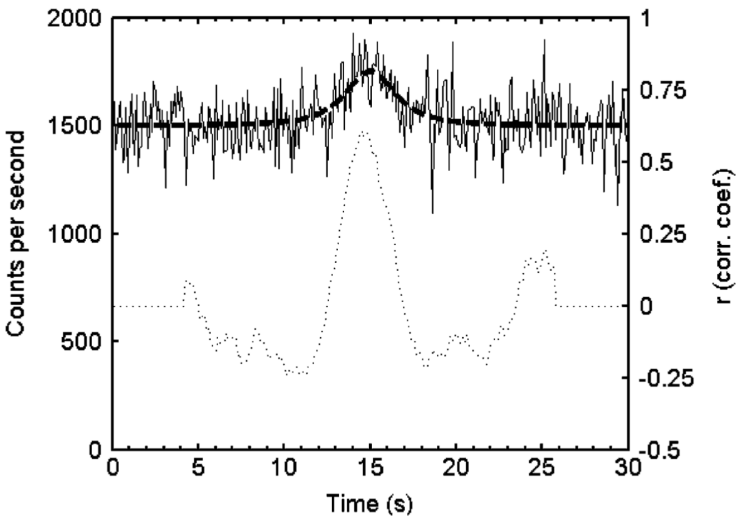


Figure 4: Figure 5 from [23] showing a count rate with injected source (dashed line), Poisson-sampled data (solid line), and correlation coefficient for the matched filter (dotted line).

shielding was placed between two detectors to induce a time-dependent signal [28]. As with the matched filter approaches, only a region of the spectrum is typically used, adding some sensitivity for particular isotopes.

### 2.2.3 Energy-domain anomaly detection and identification methods

There is a large class of algorithms that perform anomaly detection and/or identification for a single spectrum at a time, that is, considering only the current spectrum's shape and without considering recent measurements. In general, these algorithms rely on the training data set to develop a model of the background and then measure the deviation between the measurement and the model. This deviation is used as the anomaly statistic, which is usually some form of log likelihood ratio. Identification in general is performed by defining different deviation metrics, one for each kind of anomaly being considered. Although time evolution is not considered explicitly, time plays an implicit role in these algorithms in the form of the integration time (e.g., 1-second spectra), which is an important factor in determining their sensitivity.

**Region of Interest (ROI) and Censored Energy Window (CEW)** The Region of Interest (ROI) algorithm [29] and more general Censored Energy Window (CEW) algorithm [30] are methods that compare counts inside a spectral window(s) with a background estimate derived from the spectrum outside that window(s) (e.g., Figure 5). Both methods are similar to gross counts $k$-sigma in that the ROI/CEW counts are assumed to follow a Gaussian distribution with some mean and variance, but the mean and variance of the signal are estimated from the spectrum itself, not from prior training data. In addition, by focusing on certain regions of the spectrum, ROI and CEW are designed to be sensitive to specific isotopes and therefore to provide some measure of identification.

The simplest version of the ROI algorithm consists of one "source" spectral window and two "background" spectral windows, of equal width to the source window and usually adjacent or nearly adjacent on either side. For a measured spectrum, the counts in each bin are calculated. The background in the source window is estimated to be the average of the two background counts, and the variance is the square root of this estimate. More generally, the ROI algorithm can have multiple source and background windows of different widths, and regression of a training data set is used to fit the coefficients and offset to maximize the test statistic when a source is present and force it to have a mean of 0 when the source is not present.

The CEW algorithm [30] generalizes ROI in the following way. The source window becomes a binary vector (elements are 0 or 1) and the background windows become a non-negative real vector defined so that it is zero at all the spectral bins where the source window is 1. The source counts are the dot product between

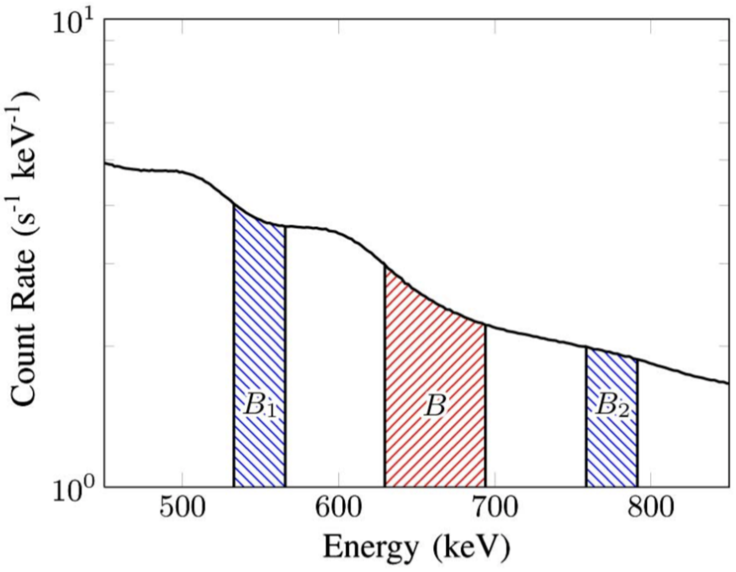


Figure 5: Figure 3 from [29] showing three ROI windows chosen to detect and identify the $^{137}$Cs photopeak in the central window at 662 keV by using the outer windows to estimate the background in the central window.

the source vector and the measured spectrum, and the analogous dot product produces the background estimate. Training data with and without the source of interest present are used to optimize the source and background windows — the source signal is maximized for data containing the source, while the mean difference between the source and background counts is zero in data collected when the source is absent.

**Matched filters in the energy domain** Similar to the matched filters in the time domain (Section 2.2.2), matched filters are sometimes used to detect and identify particular isotopes in the energy domain alone. As described in [31, 30], the matched filter can be learned from labeled training data to extract a maximal SNR signal for each desired isotope. The test statistic is the dot product of the filter with the spectrum weighted by a covariance matrix obtained from the training data.

**Principal Component Analysis (PCA)** Principal Component Analysis (PCA) has been used to detect anomalous spectra in radiation portal monitor (RPM) data [32, 33]. The PCA method learns the dominant modes of variability across the full spectrum and transforms the spectra into a subspace where anomaly detection is performed. Although this method has predominantly been applied to RPM data in the published literature, the method may also be useful for mobile spectrometer data and shares common features with other full-spectrum methods.

In [33], the training spectra were first divided by the maximum counts in any bin to normalize by intensity, making this method insensitive to gross count anomalies and only to spectral shape anomalies. The bin-wise means and variances of the intensity-normalized spectra were calculated and used to transform the data further to have bin-wise means of 0 and variances of 1. The correlation matrix of the resulting training data was diagonalized using PCA, and the first $k$ components were retained, where $k$ is selected to optimize the sensitivity of the algorithm. The spectra to test for anomalies were transformed into the principal component space, and the Mahalanobis distance between the transformed points and the mean of the training data was used as an anomaly metric. Figure 6 shows various spectra from the testing set transformed into principal component space. Notable is that the background spectra cluster around the origin, while different types of anomalies are clustered farther away. The proximity to a cluster in the transformed space can be used for anomaly identification [34].

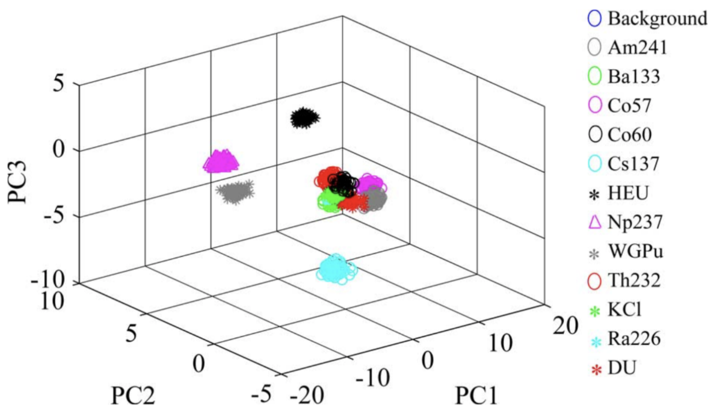


Figure 6: Figure 5 from [33] showing the testing data transformed into the space spanned by the first three principal components. The background spectra are clustered around **0** while various anomalies form clusters away from the origin.

**Linear Discriminant Analysis (LDA)** Linear discriminant analysis (LDA) is a mathematical method that produces the optimal linear solution for determining a classifier between two sets of data of any dimensionality whose variability is described by normal distributions with equal covariance matrices. As a radiological detection algorithm, LDA can be trained to classify a set of background spectra from a set of anomaly spectra, or between background and particular types of anomalies [35, 36]. LDA accomplishes this by determining a multidimensional surface that maximizes the separation between the classes while minimizing the variability within each class.

Challenges of developing representative sets for the spectral classes, the fact that LDA will over-train on non-Gaussian regions, and operational shortcomings when the variability (in each dimension or spectral bin) differs across the background and anomaly sets have hindered adaption, but the approach was considered and compared to other discriminant analysis in [37]. Other discriminant analyses such as quadratic discriminant analysis (QDA) may also be applicable as anomaly detection and identification algorithms, but tend to have more free parameters, rendering them less easy to train and continue to invoke Gaussian assumptions. Figure 7 shows a schematic from [37] illustrating how clustering is performed by LDA and QDA.

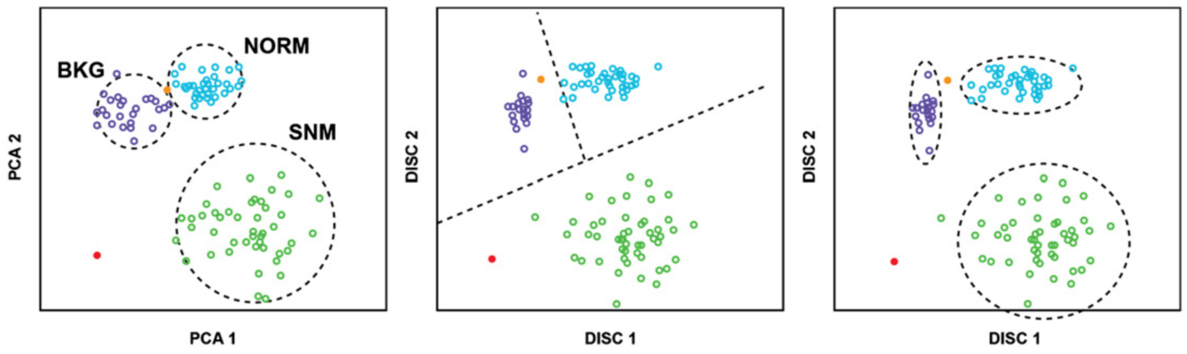


Figure 7: Figure 2 from [37] showing clusters of measured spectra projected into PCA coefficient space (left). The spectra are clustered into three groups, and LDA (center) and QDA (right) are used to draw the separations between the clusters.

**Noise-Adjusted Singular Value Decomposition (NASVD)** Noise-Adjusted Singular Value Decomposition (NASVD) [38, 39] is a spectral reconstruction method that has been used in urban search to decompose the measured spectra into linear combinations of components, some of which were background and some of which contained anomalous sources [16]. This algorithm is similar to PCA in that it transforms spectra into a $k$-dimensional subspace and examines the resulting data. One innovation suggested by [16] is to include measured anomalies in the training set so that the NASVD model learns the shape of the anomalies as well as the background. Identification can then be performed by examining the coefficients of the components that are found to strongly pattern with the introduced anomalies.

**Spectral Anomaly Detection (SAD)** Spectral Anomaly Detection (SAD) is a method where a reconstruction of the spectrum is made by projection onto an orthogonal linear subspace and back into the data space, and the squared reconstruction error is used as the anomaly metric. The linear subspace used for the reconstruction is found by using PCA to diagonalize either the covariance matrix of the training spectra or its correlation matrix, the latter of which normalizes the spectral bins by their variances and therefore improves the PCA model quality [31, 30]. SAD is a pure anomaly metric and is not able to perform identification.

**Non-negative Matrix Factorization (NMF)** Non-negative Matrix Factorization (NMF) decomposes the training spectra into a product of two non-negative matrices, one matrix containing the components and the other the weights [40, 41]. NMF is similar to PCA-based methods like NASVD and SAD in that each spectrum is decomposed into a linear combination of components, however the non-negativity constraint changes the mathematical approach. NMF has an advantage over PCA-based methods that exploit orthogonality in that it is more compatible with Poisson statistics since the Poisson negative log likelihood, rather than the sum of squared errors, can be minimized using the multiplicative update rules given in [40, 41].

NMF has recently been applied to gamma-ray anomaly detection and identification [42, 43]. In this approach, a log likelihood ratio test is performed by fitting two models: the NMF components from training, and the NMF components plus the source spectrum as an additional component. This method has been shown to have increased sensitivity over other similar methods [43]. An example of an NMF background decomposition and the fitting of an additional source component is shown in Figure 8.

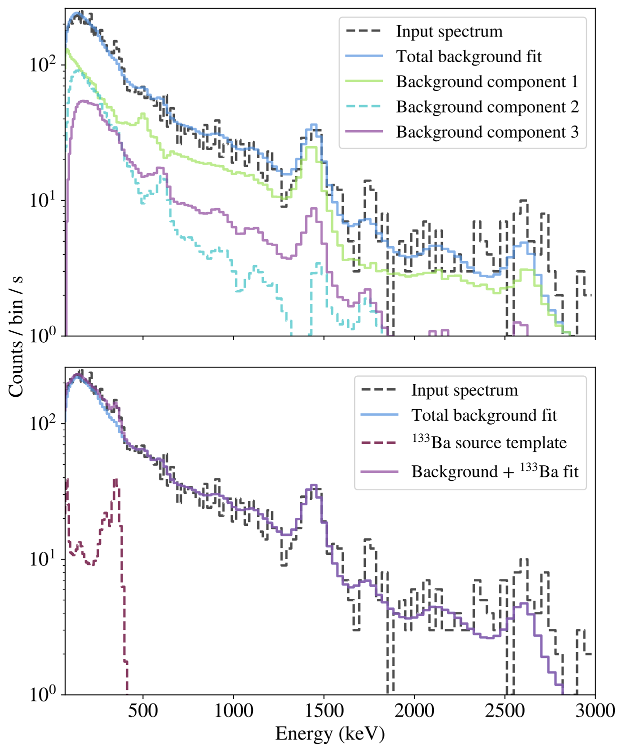


Figure 8: Figure 2 from [43] showing an NMF decomposition of a spectrum into three background components (top), and the fitting of an additional $^{133}$Ba component (bottom) to test for its presence.

**Poisson Clutter Split (PCS)** While proprietary, the Poisson Clutter Split (PCS) algorithm is described in limited detail in two publications [44, 45]. PCS uses a probabilistic representation of radiological backgrounds learned from training data, accurately accounts for Poisson statistics, and employs a generalized likelihood ratio test to perform simultaneous detection and identification of anomalous sources. For each spectrum, likelihoods are computed for the background-only and the source-present cases. The source-present tests require pre-defined normalized spectral templates. Detection and identification is performed using a likelihood ratio between these tests, combined with empirical thresholds determined from background data. The performance of PCS, compared with an ROI approach, was evaluated in [46], finding significant improvement in sensitivity for both NaI and HPGe detectors onboard a mobile detector array.

**Distribution-free approaches** All of the algorithms in this section have relied explicitly or implicitly on knowledge of the shape(s) of background spectra, but there are algorithms that make no assumptions at all about either the background or the source [47, 48]. These algorithms rely on the Kolmogorov-Smirnov (KS) test or related tests for the agreement between two distributions. The background spectrum is obtained either from recent spectra in time or from previous measurements in the same geographic location [49], and the cumulative distribution function (CDF) of the background (i.e., the fraction of counts less than or equal to a given energy) is compared to the CDF of the measured spectrum. The KS test or related test provides a statistical methodology to accept or reject the latest spectrum as anomalous or not. Such tests have no power to perform identification, only anomaly detection.

### 2.2.4 Methods using both time and energy information

Although many algorithms make some explicit use of both time and energy, there is usually sparse development in one area (e.g., using an energy window instead of gross counts) and dense development in the other (e.g., using a matched filter in the time domain). Here we highlight some algorithms that make non-trivial use of both time and energy.

**Difference Spectra** Often called “waterfall” or “rainbow” plots, two-dimensional histograms of gamma-ray events by energy and time are often used to visualize the output of a detection system. Although not an algorithm in the most technical sense, the plotting and filtering of waterfall plots illustrates some basic ideas of using time and energy simultaneously to detect anomalies. For example, in [50], the authors use a rolling average of the background spectrum as a current estimate of the background and plot waterfalls with the background removed. In addition, if the count rate of any spectrum exceeds a threshold based on $k$-sigma, that spectrum is considered a possible anomaly and is not folded into the rolling average. In this way, anomalies like $^{137}$Cs were shown to be filtered out from the variable background (Figure 9).

**Spectral Comparison Ratios and N-SCRAD** The use of spectral comparison ratios (SCRs) was proposed as a way of performing anomaly detection based on the spectral shape using a small number of coarse energy windows [51, 52]. To construct an SCR, the ratios of measured counts in the windows are compared with ratios of background estimates for the windows. The SCR method was developed into the current Nuisance-Rejection Spectral Comparison Ratio Anomaly Detection (N-SCRAD) algorithm [53, 54, 19, 55, 56]. In the time domain, N-SCRAD uses an exponentially weighted moving average (EWMA) of previous measurements to estimate the current background and its covariance. Early versions of N-SCRAD used Kalman filters to track the background and its covariance [19]. Another feature of N-SCRAD is that it can be configured to minimize signal effects due to variability in nuisance spectra, typically the three KUT backgrounds [57], but potentially due to the presence of radiologically anomalous material that is not of interest to the operator. N-SCRAD is trained to be sensitive to several anomalies at a time and thus offers anomaly detection but only indirect identification. Examples of coarse spectral windows that may be used for different groups of sources are shown in Figure 10.

**Spectral and Time Filtering** Another application of SCRs has been to combine them with wavelet filtering in the time domain to identify anomalies in RPM data [58]. In that work, spectra from RPMs were transformed into 8-bin SCRs, and the Euclidean distance ($L_2$ norm) was used as a distance metric between each measurement and the background (collected before the vehicle passed the RPM). A “Mexican Hat”

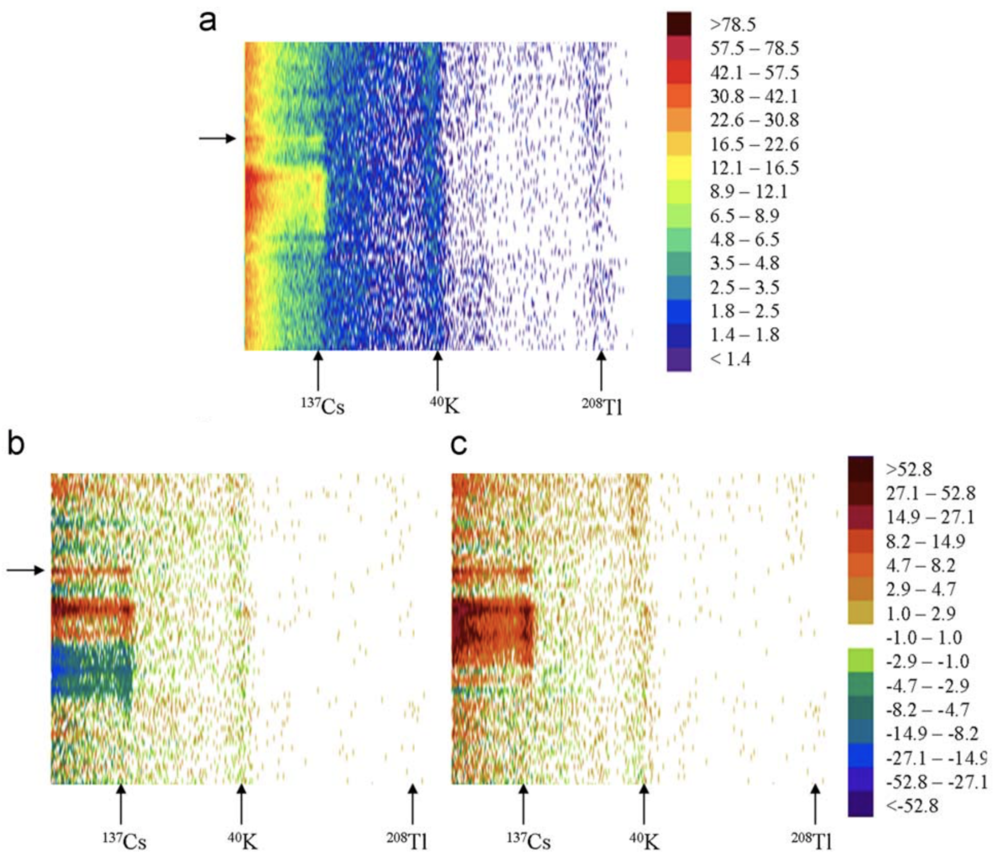


Figure 9: Figure 4 from [50] showing a waterfall plot (a), a waterfall plot where the rolling average has been subtracted (b), and the same waterfall where likely anomalies have been filtered out before calculating the rolling average (c). The final result is that $^{137}$Cs anomalies, including both the photopeak and downscatter, can be seen more clearly. The arrow indicates a brief pass by a $^{137}$Cs source that is separate from the longer pass.

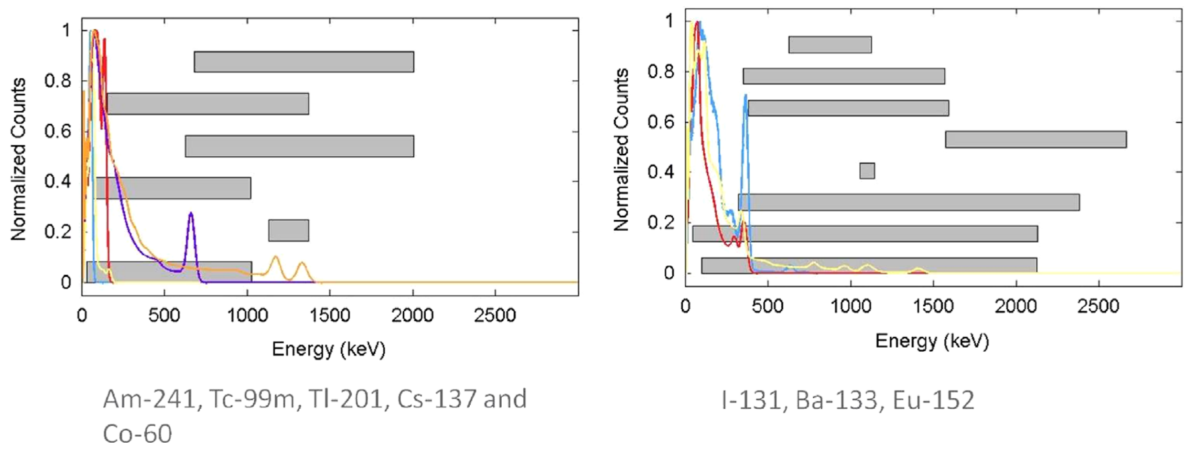


Figure 10: Figure 1 from [57] showing examples of N-SCRAD spectral windows used for detecting two groups of anomaly isotopes.

wavelet filter was then applied to the distance metric time series data to extract signals that approximate a source passing by the detectors. In combining spectral and time information, the algorithm outperformed both the spectral-only implementation and gross counts over a threshold.

**WAVRAD** WAVRAD [20] is a method that uses a continuous wavelet transform to perform variance reduction of the measured spectra. The time-domain aspect of WAVRAD is similar to that of N-SCRAD, in that a running average of recent background spectra is required to compare with the current measurement. Typically WAVRAD uses measurements from the previous 30–60 seconds to estimate the average spectral shape, which is then scaled to the current measurement's gross counts. Using the background estimate, WAVRAD calculates the deviation between the current, CWT-smoothed spectrum and the background estimate, and anomaly identification is then performed using the regions of the spectrum with the greatest deviations. WAVRAD has the additional distinction of being used in this data competition as a baseline algorithm, so a direct comparison to the competitors' results is available.

### 2.2.5 Identification algorithms

The identification of isotopes in gamma-ray spectra is an entire field of study in its own right, i.e., separate from the detection problem, and the literature goes back decades. Some detection techniques that include identification or the potential for identification have been discussed in previous sections, and the following section on neural network approaches will also mention some relevant identification algorithms. For a literature review focused only on identification, the reader is directed to a recent review of the existing literature in [59].

### 2.2.6 Neural Network-based approaches

Neural networks have recently become the state of the art methods for complex data analysis, especially for object detection, identification, and segmentation in images. They also find wide spread applications in other complex data streams such as natural language processing, fraud detection, and machine diagnostics. Despite their tremendous success in these areas, research in their application to radiation detection, identification, and localization has only recently begun. This section will give a brief overview of this research.

**Autoencoders** There are many different neural network architecture designs that are used for a variety of different applications. One such architecture is the autoencoder, which is a type of neural network composed of two primary parts: the encoder and the decoder. The autoencoder is normally trained in an unsupervised fashion, that is, the input data is the same as the target output data. The encoder is a series of layers (densely connected, convolutional, locally connected, etc.) that compresses the input data into a layer called the latent space which normally has a significantly lower dimensionality than the input space. The decoder then takes the output of the latent space and through another series of neural layers, expands it back into its original dimensionality that is then considered the output of the network. Through training, the encoder must learn to identify salient features in the input data such that the decoder can reconstruct the input data from the dimensionally-constrained latent space.

One example is a neural network architecture for radiation anomaly detection called the Autoencoder Radiation Anomaly Detection (ARAD) algorithm [60]. ARAD is targeted at the source search problem in dynamic background radiation environments and is trained on a set of gamma-ray spectra that represent the typical radiation background in a given search environment. When source-containing spectra are input into the trained network, the reconstruction at the network's output is only able to reconstruct the background components but not the source components, leading to a spike in reconstruction error that can trigger an alarm. One of the main causes of false alarms in source search (especially urban) is the highly dynamic nature of the radiation background. The main contributors to this dynamic background are variations in the absolute and relative concentrations of NORM in different materials [61, 62, 63, 64, 65], physical clutter that can attenuate and scatter photons [66, 67], and precipitation-induced wet-deposition of $^{222}$Rn and its daughters $^{214}$Bi and $^{214}$Pb [68]. By including data from these dynamic background contributors in the ARAD training set, ARAD is able to recognize and reconstruct these components in collected spectra, reducing the likelihood of alarming on NORM fluctuations. Like many of the anomaly detection approaches described above, one of the advantages of autoencoder-based anomaly detection is that the model does not require any source data to make a detection decision, allowing it to more rapidly be deployed for a real world measurement campaign. The initial version of ARAD was published and presented at the 60th Annual INMM meeting [69]. The algorithm is described in depth in [60] and an additional manuscript is in preparation [70]. Figure 11 shows an example of the ARAD reconstruction error for a run generated from the data competition model of Chameleon street (see Section 3) with an $^{131}$I source injected at a strength producing a peak signal to noise ratio of 9 in the gross count rate in the detector. From this figure we can see that the background count rate varies significantly over the course of the run, with a magnitude and rate of change rivaling that from the injected source. Despite this, the ARAD reconstruction error overcomes the alarm threshold at the source but stays well below alarm level for the rest of the run. This alarm threshold was calculated by setting a false positive rate of 1 per 8 hours. Figure 12 shows one of the alarm spectra resulting from the $^{131}$I source along with the ARAD reconstruction of this spectrum. As can be seen, ARAD was able to reconstruct the background components of the spectra but was unable to reconstruct the 364 keV photopeak emitted by the $^{131}$I source.

Autoencoders have also been used outside of the source search problem. An autoencoder was designed and developed to reconstruct the Compton edges from noisy PVT gamma-ray spectra [71]. This autoencoder is trained on a set of Monte Carlo synthetic data where the input data are gamma-ray spectra with Gaussian energy broadening (GEB) applied and the target output data are the same spectra without GEB. When trained on these data, the autoencoder learns to extract a mapping between the GEB and non-GEB spectra.

**Convolutional Neural Networks** At least one study has explored the application of supervised 2D convolutional neural networks (CNN) designed for image processing to the radioisotope detection and identification application space [72]. This particular study used an ensemble of three common neural network architectures: VGGNet [73], Inception [74], and ResNet [75] to classify non-background radioisotopes (or the lack thereof) in time series images of gamma-ray spectra rendered as "waterfall" plots. These networks were trained on the initial data set created for the government data competition and then tested on real world data collected at the Northern Virginia Array (NoVArray) [76]. The anomalous radiation events contained in the NoVArray data set are dominated by medical radioisotopes along with a smaller selection of industrial radioisotopes and special nuclear material [76, 72]. The conclusions from this study demonstrated that CNNs trained on simulated data can effectively be used on measured data as well without a significant negative impact on detection/identification performance [72]. Also demonstrated is a method for ensembling the outputs of each CNN in order to make a single detection/identification decision. It is important to emphasize that the work presented in this report enabled the development of this algorithm, which highlights the significance of our work to furthering the state of the art in radiation detection and identification algorithms, particularly for data-dependent algorithms such as neural networks.

Another group of researchers developed a novel CNN architecture that operates directly on the 1D gamma-ray spectral histograms generated from CdTe detector data to perform radioisotope identification [77]. This model consists of a collection of parallel multilayer CNNs (one for each radioisotope of interest) which are trained on purely synthetic data generated using the Monte Carlo–based Geant4 library. Each CNN contains two output neurons — one for source present and the other for source not present — and is run independently on the input spectrum in order to detect a single radioisotope in the input spectrum. This parallel network approach is unconventional when compared to most neural networks designed for classification tasks which normally have a single network with multiple output neurons each corresponding to a particular class. One potential advantage of the parallel independent model is that the entire network does not need to be retrained every time a new radioisotope is added to the library. On the other hand, the computational requirements for such a system will likely increase far greater than if one was to add a single extra neuron to the output as is traditionally done in classification networks. The results of this study indicated good detection/identification performance when applied to real data taken in laboratory conditions for six different radioisotopes [77].

Another study performed a comparison between a shallow fully-connected neural network and a relatively shallow CNN on the radioisotope identification problem in NaI(Tl). The models were trained on synthetic data representing 1D gamma-ray spectral histograms for 29 different sources and a uniformly distributed background spectrum [78]. The results of the study demonstrated better performance using the CNN rather than the fully-connected network, however both methods were sensitive to perturbations in the background radiation and source-to-detector standoff distance [78]. The authors indicated that these problems could

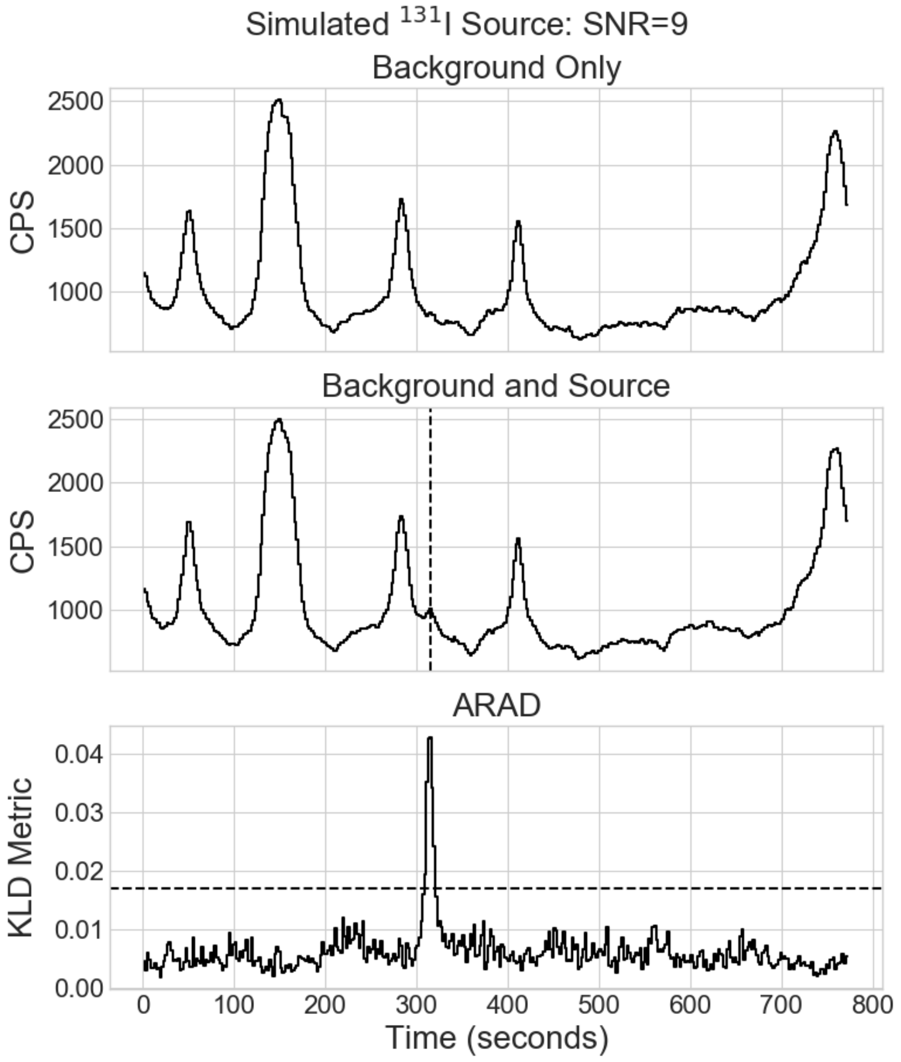


Figure 11: Countrate and ARAD reconstruction error for a simulated run generated from the data competition model of Chameleon Street along with an $^{131}$I source injected at a strength producing a peak signal to noise ratio of 9 in the gross count rate in the detector. The background-only countrate is shown in the top plot of the figure, highlighting the dynamics of the radiation background as the detector moves throughout the scene. The middle plot shows the countrate from both background and the source, with the dotted line at the point of closest approach between the detector and source. Finally, the bottom plot shows the ARAD reconstruction error for this run with a dotted line indicating the alarm threshold (set for a false positive rate of 1 per 8 hours), generating an alarm at the correct position.

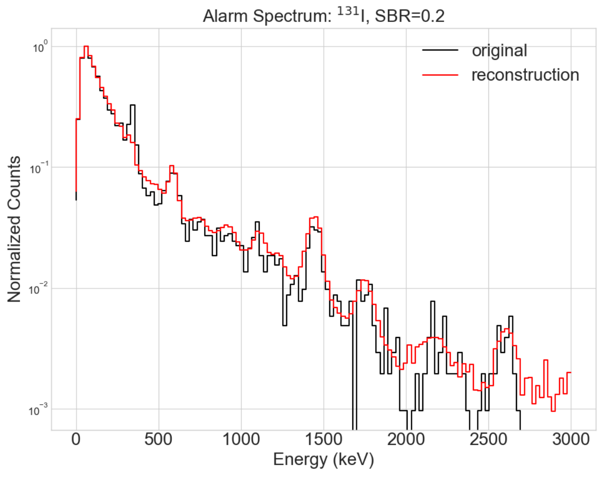


Figure 12: ARAD reconstruction of a gamma-ray spectrum collected during the alarm phase for the run in Figure 11. This spectrum generated a detection alarm because ARAD was unable to reconstruct the 364 keV photopeak from $^{131}$I.

likely be solved by expanding and modifying the training data set [78].

**Transfer learning** The work described in [72] represented a second novel approach to neural network application to radiation detection problems. The authors leveraged the simulated data from the government competition to train a CNN. They then applied transfer learning wherein a subset of CNN's layers were retrained on real labelled data.

**Perceptrons and Other Elementary Networks** Research into the use of neural networks for radioisotope identification in gamma-ray spectra began at least as early as 1992 [79]. These earlier efforts mostly focused on the use of shallow non-convolutional neural networks. One study in particular used a single layer neural network architecture called an auto associative memory to quantify and identify a variety of radioisotopes in mixed-sample gamma-ray spectra [79]. This network was trained on source spectra taken by an NaI(Tl) detector in a laboratory using Greville's algorithm for network optimization [79], as opposed to back-propagation which soon became the algorithm of choice for training neural networks.

Single layer perceptron networks have also been used in combination with engineered feature transformations for radioisotope identification in HPGe spectra [80]. Given that single layer perceptrons can only learn to extract linearly separable features in the data in which they are trained, it was common practice to use engineered feature transformations to project the input data onto a more linearly separable space. In this particular study, the researchers used singular value decomposition on a pre-defined set of energy ROIs to project the inputs onto an orthogonal space prior to input into the single layer network [80]. The results of this work were promising and offered a viable alternative to spectral deconvolution, another method for radioisotope identification in spectra [80].

## 2.3 Summary

In order to leverage the information contained in list-mode gamma-ray detector events, many types of algorithms have been developed and deployed over the years to discriminate between background and non-background environments and to identify the cause of any non-background conditions. From the simplest algorithm of monitoring gross counts in a spectral window to the decomposition of spectra into linear components, to the training of CNNs to find anomalies in 2D waterfall histograms, all these algorithms have a commonality of trying to determine useful filters in both the time and energy domains to improve anomaly detection and identification performance for a detector system. Some focus exclusively on the time domain, trying to leverage local count rate behavior, while other focus exclusively on the energy domain, often leveraging the entire spectral shape to find anomalies, and yet others apply filters in both time and energy to attack the problem.

Linear algebra and Gaussian statistics have been the workhorses in this domain for years, but recently the emerging field of deep learning is beginning to have a major impact, and given the complexity of the problem it is likely that deep learning will lead to many advances in the coming years. This trend towards data-dependent algorithms such as deep learning highlights the need for high-quality, large, and realistic labelled data sets to fuel future innovation in these areas. This data competition demonstrates the ability to generate such data and use it for the development of novel methods for radiation detection and identification. This is especially highlighted by the recent development of two deep learning-based algorithms for radiation detection and identification. One of these is the recent state-of-the-art work of [72], where the data set presented in this report was used to pre-train a 2D CNN on gamma-ray waterfalls. The network was then retrained on data from a real detector system with labeled anomalies and proved to perform very well in such an environment. The data competition data set was also used to both design, develop, and test the autoencoder radiation anomaly detection (ARAD) algorithm, funded through the Department of Homeland Security, which has been tested and proven for use in a real-world detection scenario [60, 69, 70]. These accomplishments demonstrate that this data set is already having an impact in the real world by enabling new types of algorithms that can work with real detector data, and it highlights the role that large, high quality synthetic data sets can play in training these new kinds of algorithms.

# 3 Radiation Transport Simulations

Gamma-ray background is composed primarily of potassium, uranium and thorium (KUT) present in everyday materials such as brick, concrete, and asphalt [81, 82]. Photon fluxes due to KUT are highly variable in an urban search environment due to the fact that nearby buildings can emit wildly different background signatures [20]. The presence of objects in the scene is also very important in measured background detector response. Cars parked along streets, and other forms of 'clutter', provide shielding for gamma rays emitted from sidewalks and nearby buildings [67]. Because of these factors, it is very challenging to predict the gamma-ray background response to a high degree of accuracy at an arbitrary location and the background signal may fluctuate strongly as a detector moves through an environment.

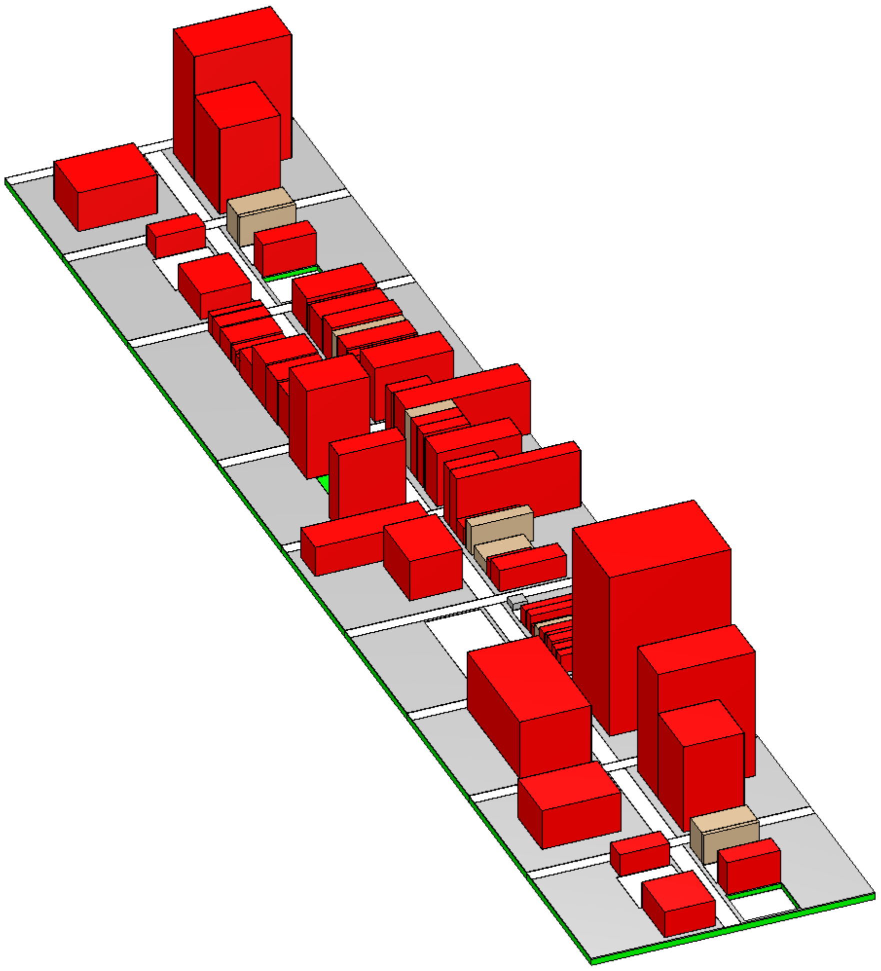
This work describes the creation of a virtual test bed, based on full 3-D Monte Carlo radiation transport simulations, with realistic gamma-ray backgrounds and simulated sources. Other efforts have generated radiological source terms using experimental data, Monte Carlo simulations [7], detector response simulation codes (like GADRAS) [8] and/or solid angle calculations [9]. These source terms are usually injected into a measured background or a simple model [10]. This work differs from other efforts in that the backgrounds are generated using Monte Carlo radiation transport simulations, meaning the background composition and variability can be easily modified. Having a diverse set of background variations is important considering the difficulty in detecting weak sources compared to background.

The virtual test bed consists of a collection of data sets representing time series detector response data for a 2-in×4-in×16-in NaI(Tl) detector moving through a city street without cars or other forms of clutter. Clutter was not included in this test bed to simplify the model generation and simulation processes. In the virtual test bed, background composition and magnitude can be varied between data sets, and localized sources of interest can be placed at any location. Because the data is based on a model and everything in the scene is known, high-quality labels can be applied to each synthetic data set to train and evaluate algorithms. High quality data labels are very important when training and evaluating data driven algorithms, especially those which require large data sets to train. Moreover, multiple algorithms can be evaluated on a common data set to understand [relative] algorithm performance in a variety of conditions such as detector speed, background variability, source type, and intensity.

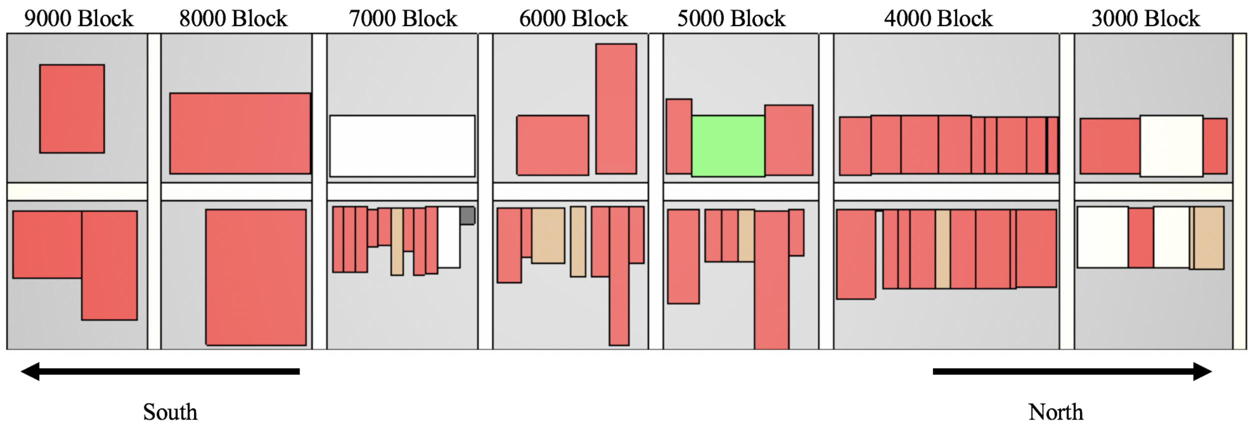
Because modern data driven algorithms require large, diverse data sets to train and test on, a fixed geometry with fixed background source strengths is not sufficient. To increase data diversity, Monte Carlo calculations used a set of interchangeable city blocks, so that many different 'instances' of the model could be used to create different streets. Gamma-ray background due to the different KUT components within each block were computed separately, allowing them to be added back together with different strengths when generating synthetic data. This model was called 'Chameleon Street' due to the ability to change the order of the blocks and change the strength of the KUT concentrations in the background components, which increases the variability in gamma-ray background throughout the model

## 3.1 The Basic Model

Chameleon Street consists of a set of seven city blocks, based on the 3000 to 9000 blocks of Gay Street in Knoxville, TN, each with a level asphalt center street (40 ft or 12.192 m wide), a smaller asphalt side street (30 ft or 9.144 m wide), and several buildings. Figure 13 (a) shows a 3-D rendering one instance of the model and Figure 13 (b) shows seven blocks of the model with labels. Both are arranged in an order similar to Gay Street. The buildings are hollow shells with a wall thickness of 6 in (15.24 cm). They are made from concrete, granite or brick. Building heights are mostly less than 45 m with the tallest at 125 m. Asphalt parking lots and soil areas appear in some of the blocks. Only five materials and air are used in the model. Material compositions were taken from a PNNL report [83]. The different blocks all align on the center street and can be placed in any order. The blocks have different lengths down the center street, ranging from 88.4 to 143.3 m. When stacked in different orders, the length of the street is always the same (786 meters), however the side streets do not appear at the same distances from the ends. The base concrete layer serves as the floor of each building and the sidewalks (and other areas) outside of the buildings. The streets are 6 in (15.24 cm) below the sidewalks. Parking lots match the street level except the one in the northeast corner, which is recessed almost five meters below street level. Table 1 shows the order of the blocks for the eight different 'instances' (a given order of blocks) used in the data competition.



(a)



(b)

Figure 13: (a) A 3-D rendering of model instance 1 of Chameleon Street. Concrete is gray, brick is red, granite is tan, asphalt is white, and soil is green. Note that the first and last blocks are repeated on the opposite ends. (b) An overhead view of seven basic blocks of Chameleon Street with each block labeled. The arrangement matches Gay Street in Knoxville, starting with the 9000 block on the left (south) and ending with the 3000 block on the right (north).

Table 1: The orders of the Chameleon Street blocks used in each model instance in the data competition.

| | Instance | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| North→ | 3000 | 8000 | 6000 | 3000 | 3000 | 6000 | 8000 | 5000 |
| | 4000 | 3000 | 5000 | 9000 | 7000 | 7000 | 5000 | 8000 |
| | 5000 | 7000 | 4000 | 8000 | 9000 | 8000 | 6000 | 9000 |
| | 6000 | 4000 | 3000 | 7000 | 5000 | 3000 | 3000 | 6000 |
| | 7000 | 9000 | 7000 | 4000 | 4000 | 5000 | 7000 | 7000 |
| | 8000 | 5000 | 9000 | 6000 | 6000 | 4000 | 4000 | 3000 |
| ←South | 9000 | 6000 | 8000 | 5000 | 8000 | 9000 | 9000 | 4000 |

For a given instance of the model, after the order of the seven blocks is determined, the first and last blocks are repeated on the opposite ends. The final models were 201 m wide, 989-1047 m long (depending on instance) and 158 m high (5.5 m below the street, the rest above). Tallies are only made in the 786 meters of the original seven blocks, but the presence of the repeated blocks effectively gives the model periodic boundaries, so passes can start at any place in the geometry and 'wrap around' to the other end. Randomizing starting positions and having multiple block orders was done in part to make it difficult if not impossible for competitors to infer any global information about the Chameleon Street model.

## 3.2 Background Components

The biggest source of background radiation in an urban environment is the NORM contained in the roadways, sidewalks and various building materials. The components of NORM used in the data competition were $^{40}$K, $^{232}$Th and its daughters, $^{238}$U/$^{235}$U and their daughters. Cosmic radiation was not considered because it only contribute to about 20 counts/sec in a 2-in×4-in×16-in NaI(Tl) detector, or about 1-2% of the total count rate in a typical urban environment [20]. Background for a given model instance was separately simulated for each KUT component of each material (asphalt, brick, granite, concrete, and soil) for each block. Granite is contained in only five of the blocks. With the three main NORM components in each of the 27 material/block combinations, plus the cesium in the soil, there are a total of 82 background components for each model instance. Each of the 82 background components was simulated with a concentration of 1 Bq/kg of the specific NORM component, so they could be combined later with any given activity concentration. The background components were computed for each instance of Chameleon Street to ensure the that scattering was correctly modeled from the different neighboring blocks.

## 3.3 Sources

For each of the separate blocks, two source locations were chosen (three for the 4000 block) for a total of 15 locations in five different types of environments. When the blocks are combined in different orders, those locations will appear at different positions along the length of the street and the gamma-ray fluxes in the street will be influenced by the neighboring blocks (scattering). Figure 14 shows the source locations, and Table 2 lists the distance of each source from the centerline of the street. Sources were 100 centimeters above the sidewalk, except for location 32, which was 3.4 meters below the sidewalk level, in the recessed parking lot. Because the blocks appear in different orders and different sources appear at the 15 source locations, it was deemed unlikely that competitors or their algorithms would be able to infer and leverage the finite number of source locations.

A small set of five sources was chosen for the data competition - two special nuclear material sources (SNM), highly enriched uranium (HEU), weapons-grade plutonium (WGPu), and three common isotopic sources, $^{99m}$Tc, $^{131}$I, and $^{60}$Co. The geometry and composition of the SNM sources are IAEA significant quantities [17]. Two variants of each source were simulated: (1) the source itself and (2) the source with 1 cm of lead shielding. Sources were simulated at nominal strengths that could later be scaled up or down when they were injected into the competition data. Table 3 contains some specific details about the nominal strength simulated sources.

Table 2: Source locations covered five different environments in all of the blocks.

| ID | Environment | Distance from road centerline (feet) | (m) |
|---|---|---|---|
| 32 | parking lot (recessed) | 50 | 15.24 |
| 71 | parking lot | 100 | 30.48 |
| 41 | front of building | 37 | 11.28 |
| 92 | front of building | 80 | 24.38 |
| 43 | interesting corner | 35 | 10.67 |
| 72 | interesting corner | 35 | 10.67 |
| 51 | interesting corner | 40 | 12.19 |
| 31 | side of building | 50 | 15.24 |
| 62 | side of building | 60 | 18.29 |
| 81 | side of building | 70 | 21.34 |
| 82 | side of building | 80 | 24.38 |
| 91 | side of building | 100 | 30.48 |
| 42 | between two brick buildings* | 40 | 12.19 |
| 61 | between two granite buildings | 50 | 15.24 |
| 52 | between two brick buildings | 60 | 18.29 |

*Source location 42 is between two brick buildings and behind a brick column.

Table 3: Sources (bare and with 1 cm of lead shielding) simulated at each source position in each model instance.

| ID | source | Z | Description |
|---|---|---|---|
| 1 | HEU | 92 | Highly enriched uranium<br>25 kg, 90.3% $^{235}$U<br>18.95 g/cc, aged 20 years |
| 2 | WGPu | 94 | Weapons grade plutonium<br>8 kg, 93.63% $^{239}$Pu, 6% $^{240}$Pu<br>19.86 g/cc aged 20 years |
| 3 | $^{131}$I | 53 | Iodine, a medical isotope<br>1 $\mu$Ci, point source |
| 4 | $^{60}$Co | 27 | Cobalt, an industrial isotope<br>1 $\mu$Ci, point source |
| 5 | $^{99m}$Tc | 43 | Technetium, a medical isotope<br>1 $\mu$Ci, point source |

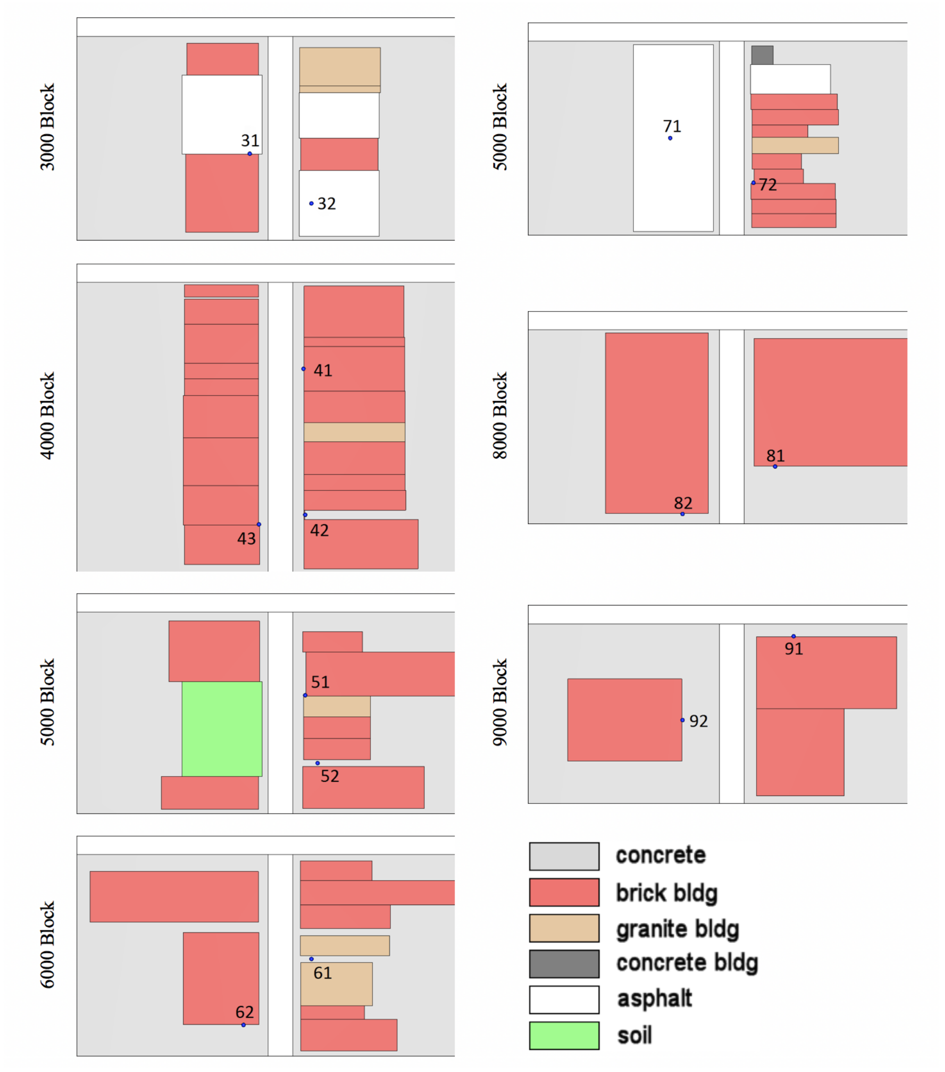


Figure 14: Locations of the simulated sources, using an ID number corresponding to the block.

## 3.4 Simulations

Separate transport simulations in MAVRIC [3] were run for eight instances of the Chameleon Street model, each with 82 background components (27 physical components and three isotopes plus cesium in soil) and 150 simulated sources (five source types, two levels of shielding, and 15 locations). Using this methodology, any background model could be paired with any source in post-processing. To reduce the overall computation time, the simulations used advanced variance reduction techniques to optimize the calculation of the mesh tallies in the street. The FW-CADIS method was used with the goal of obtaining low relative uncertainties in each bin of the tallied fluxes along the lanes in the street [84]. For the background calculations, the biased source preferentially sampled gamma rays near the surface of the road and the front surfaces of the buildings. For both background and source simulations, the importance map preferentially killed off low-weight gamma rays traveling deep into buildings or the road. These simulations would not have been attempted without the FW-CADIS variance reduction capabilities in MAVRIC.

Each component of background was run for 100 hours, and each source was run for 24 hours (MAVRIC is a serial code, so these times are for a single AMD Bulldozer 2.3 GHz processor). These times were chosen so that after processing with the detector response and energy resolution function, statistical uncertainties from the Monte Carlo were small enough not be noticable in the final synthetic data. These spectra represent the spectra that would have been seen for a very long dwell time at each mesh voxel location.

## 3.5 Tallies and Detector Response Function

Four mesh tallies, each corresponding to the centers of the four travel lanes along the center roadway, were used in every simulation. The centers of the voxels were 1.5 m above the sidewalk level. Background simulation mesh tallies used 100 cm cubic meshes covering the 786 meters. The simulated sources used a more tailored mesh to reduce file size and better show the sharp changes in count rate near the source. Within 150 meters of the source position along the street, 100 cm cubic mesh voxels were used. Within 25 meters of the source, the mesh cells were reduced to 25 cm meshes in the direction of the main road because the simulated source gamma-ray flux varies strongly near the point of closest approach. The remaining dimensions of these voxels were still 100 cm. For sources located in blocks that were repeated (near the ends of the model), the mesh tallies were placed near both simulated sources, to preserve model periodicity.

Tallies recorded the gamma-ray flux in each mesh voxel using 1 keV energy bins from 10 keV to 3000 keV (2990 bins). The energy-dependent flux in each mesh cell, an example of which is shown in Figure 15 (a), was combined with a detector response function created for gamma rays striking a 2-in$\times$4-in$\times$16-in NaI(Tl) detector to create a pulse-height count rate spectrum. An energy resolution function from GADRAS [85] was also applied to smear the pulse-height spectra into the final count rate spectra. See Figure 15 (b) for an example NaI(Tl) count rate spectrum.

The accuracy of the modeling and detector response generation methodology has been studied using a radiation transport test bed of the Fort Indiantown Gap National Guard facility in Pennsylvania [86]. Modeled detector response functions, for both background only and source plus background simulations, have been compared with experimental data [87].

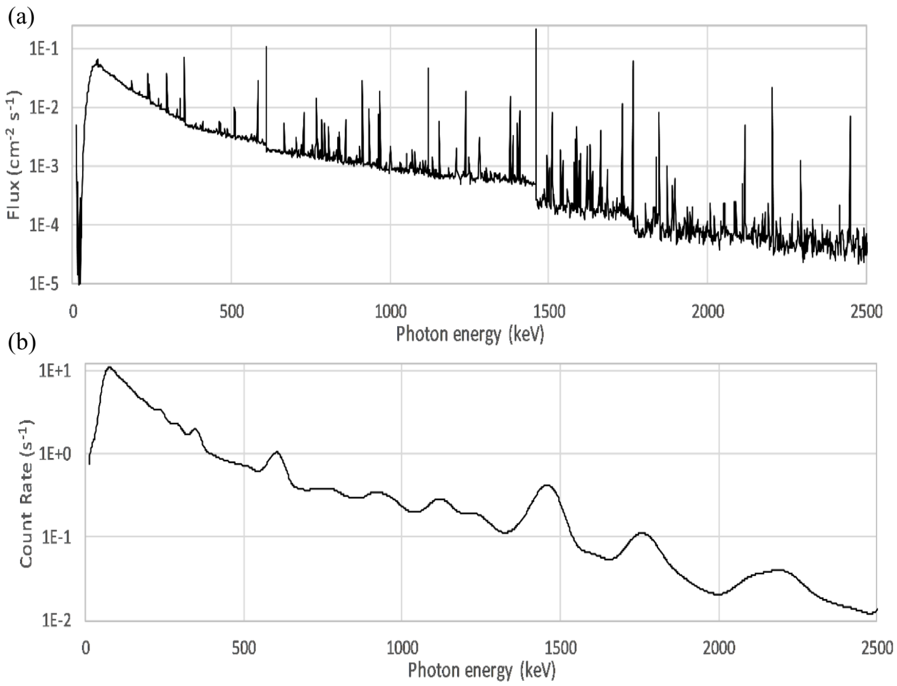


Figure 15: (a) Example gamma-ray flux mesh tally and (b) energy broadened pulse-height count rate spectrum. Energy bins are 1 keV wide.

# 4 List-Mode Data Generation and Testing

In order to give users maximum flexibility in organizing the data in any time and energy structure, list-mode data (a list of timestamps of when a gamma-ray detection occurred along with its associated energy) was provided. Each data set represents the response of a detector moving at a constant speed down a single lane of traffic due to background and potentially to simulated source emissions. Background and simulated source (if present) data sets were generated separately and combined at the end to allow for flexibility in synthetic data creation.

## 4.1 Background Data Generation

To generate background list-mode data sets, a set of model parameters were chosen including detector speed, starting location, ending location, lane of travel, background model, and KUT background activities ($b_{jkl}$ over isotope index $j$, block index $k$ and material index $l$) for each material in each block of the model were used to generate a set of detector responses ($B_i$) in 1 meter voxels (voxel index $i$) through the detector path. For the TopCoder data competition, parameters for each data set were chosen so that the model parameter ranges spanned the space efficiently, using a non-uniform space filling approach which aims to focus the parameter space in a challenging region for data competition participants as discussed in Sec. 6.2 and Refs. [88, 5, 6]. For the TopCoder competition the following constraints were placed on the model parameters:

- The detector speed was limited to 1–13.4 m/s and remained constant to span from walking to 30 mph city driving speeds.
- Each data set had to be at least 45 seconds long, limiting minimum path length for a particular speed to give algorithms enough time to collect background.
- The detector position stayed in one of the four lanes.
- KUT background activities were varied ±80% from nominal values.

KUT background activities were varied ±80% from nominal values, see Table 4 [86, 89], to simulate varying concentrations of these isotopes in real world materials and to ensure generated backgrounds did not have high correlations between K, U, and Th components while representing total count rate variations that reflect real measured data. These variations are in-line with real world variations in these materials [90, 91]. Strong correlations between K, U and Th in the data were unfavorable because algorithm developers may rely on these in developing their algorithm methodology. A correlation matrix, shown in Table 5, was computed using K, U, and Th count rates from all voxels in each model using about 10,000 sampled KUT background activities. A small positive correlation is expected in the data because of geometric effects. For instance, if the detector moves closer to a granite building, K, U, and Th components will all increase and then decrease when moving away.

Table 4: Nominal NORM concentrations (Bq/kg) used in the materials of the Chameleon Street models.

| | $^{40}$K | $^{232}$Th and daughters | $^{238}$U/$^{235}$U and daughters |
|---|---|---|---|
| asphalt | 58.52 | 3.96 | 24.34 |
| brick | 150.52 | 8.06 | 14.77 |
| concrete | 138.77 | 10.2 | 18.21 |
| soil | 247.58 | 37.75 | 25.88 |
| granite | 720 | 75 | 125 |

This approach allows for the generation of a wide variety of background detector response spectra for a single geometry. Figure 16 (a) shows the the gamma-ray detector response spectrum for each background component and the total spectrum using nominal KUT activities in model instance 1. Because the detector response for each material in each block were calculated independently, contributions to the detector response from all materials can be tracked throughout the model. Figure 16 (b) shows the total count rate (spectrum summed as function of position) from each material in each block for a detector moving through Chameleon Street instance 1 at 1 m/s. In these models, granite buildings produce large peaks in the total detector response, because granite typically has high concentrations of KUT relative to other common materials [91]. The peaks in the asphalt and valleys in the concrete detector response are due to the presence of the side streets. Using this information one may be able to understand how algorithms respond to the presence of certain types of objects in the scene, possibly in terms of false alarms or false positive rates.

Table 5: Correlation matrix between K, U and Th count rates using ±80% from nominal background variation.

| | K | U | Th |
|---|---|---|---|
| K | 1.0 | 0.436 | 0.533 |
| U | 0.436 | 1.0 | 0.536 |
| Th | 0.533 | 0.536 | 1.0 |

To further increase detector response variability and make it harder for competitors to simply "learn" the geometry of one model, eight Chameleon Street instances were used; detector starting and ending locations were varied (to vary the total length of travel); and four lanes of travel were used (the detector can move in either direction through the model). The total count rate for a detector moving through each background model is shown in Figure 16 (c), where the variation in the detector responses arises from the differences in model geometry. Because the detector response in each model are periodic along the direction of the roadway, a data set can start towards the end of the model and end near the beginning.

For a selected parameter set, the isotope (index $j$), block (index $k$) and material (index $l$) dependent Monte Carlo detector response spectra, $B^{DR}_{ijkl}$, were scaled by corresponding activities $b_{jkl}$. The total number of counts per second of a detector $B_i$ (units of counts/sec) in $i$ is

$$B_i = \sum_j \sum_k \sum_l b_{jkl} B^{DR}_{ijkl} \tag{2}$$

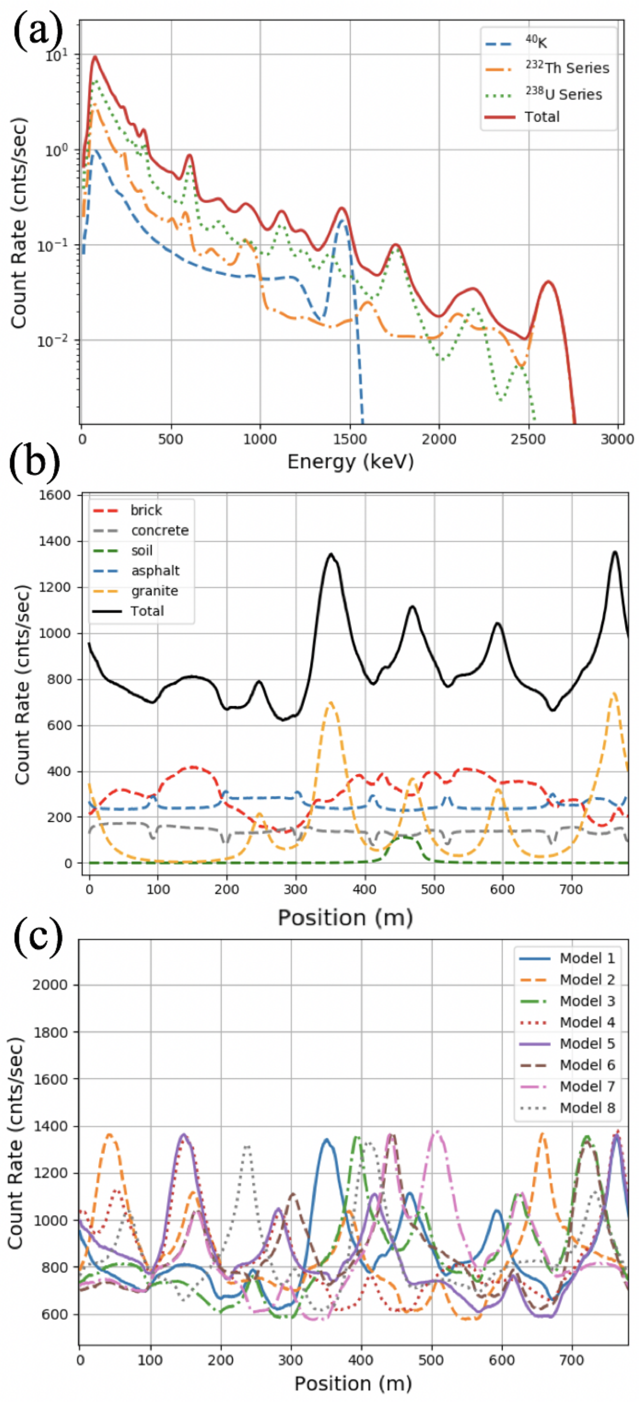


Figure 16: (Color Online) (a) Gamma-ray detector response spectrum averaged over Chameleon Street model instance 1 using nominal KUT activities found in Table 4. Energy bins are 2 keV wide. (b) Material contributions to the total count rate for a detector moving at 1 m/s through using nominal KUT activities. (c) Total count rates as a function of position for all eight Chameleon Street models.

where $B_{ijkl}^{DR}$ is in units of (counts/sec)/(Bq/kg), and $b_{jkl}$ is in units of (Bq/kg). The amount of time the detector spent in each voxel ($\Delta T_i$) was calculated by

$$\Delta T_i = v\Delta x_i \tag{3}$$

where $v$ is the detector speed in meters/seconds and $\Delta x_i$ is the voxel size in meters. $\Delta x_i$ was used to scale the count rate detector response $B_i$ in each voxel to generate the mean spectrum (units of counts). The mean spectrum in each voxel was sampled over each energy bin using Poisson statistics to generate a statistically representative detector response. To generate list-mode data, each count in a statistically representative detector response spectrum was spread uniformly in time and energy. Timestamps are assigned by sampling from a uniform time distribution with bounds beginning when the detector first enters the voxel and ending when the detector leaves. Energies are assigned by sampling from uniform energy distributions defined by the upper and lower energy bin edges of the selected channel of the spectrum. This approach assigns each detected count with a timestamp and an energy.

## 4.2 Simulated Source Data Generation

To generate simulated source data for the competition, a set of source parameters were chosen including simulated source type ($^{60}$Co, $^{99\text{m}}$Tc, $^{131}$I, HEU, WGPu and $^{99\text{m}}$Tc + HEU), shielding, location, and activity. The total count rates for the nominal activity bare HEU source at all 15 source locations in Chameleon Street instance 1 are shown in Figure 17. Because source locations have different offsets from the road, and some have different levels of environmental shielding (or shielding coming from the model geometry itself), the total number of counts reaching the detector for a the same source activity across different source locations varies widely. To focus on the source strengths that would be detectable, the signal-to-noise ratio (SNR) at the point of closest approach was calculated for each source and source location. Using the number of counts from the source, $S$, and from background, $B$, on the detector at the point of closest approach, $i = i'$, the $SNR$ is calculated using the following expressions,

$$SNR = \frac{S}{\sqrt{S+B}}, \tag{4}$$

$$S = aS_{i'}^{DR}\Delta T_i, \tag{5}$$

and

$$B = B_{i'}\Delta T_i, \tag{6}$$

where $a$ is the source activity (units of Bq), $\Delta T_i = 1$ second, and $S_{i'}^{DR}$ is the simulated source-induced detector response in units of (couts/sec$\cdot$/$\mu$Ci) or (couts/sec$\cdot$/SQ) for SNM sources. One can calculate the desired source activity, once the background component activities and desired $SNR$ were chosen, by solving for the physical value of $S$ in Eqn. 4.

Sample spectra from each source and shielding combination are shown in Figure 18. To generate source detector responses with different activities, the Monte Carlo detector response was simply scaled by a multiplicative factor. However, for SNM sources, the source geometry is noteworthy because of the effects of self-attenuation. To get realistic varying levels of activity, one cannot just scale their activities. Instead, one should generate new source terms with new source geometries. Unfortunately, running thousands of Monte Carlo SNM source geometries was too costly, so the team decided to scale SNM in a non-physical way by simply using the multiplicative scaling. For the combination of HEU and $^{99\text{m}}$Tc, the HEU response was scaled so that SNR at the point of closest approach was constant (SNR=5.0) for all runs while $^{99\text{m}}$Tc activity varied. This simplified the parameter space and is used to investigate when an algorithm would detect and identify both HEU and $^{99\text{m}}$Tc at the same time.

Once a source type, location, source activity (based on some desired SNR), and shielding were chosen, the source spectra (in units of counts) in each model voxel was generated in the same fashion as the background. These spectra were again sampled over each energy bin using Poisson statistics to generate the detector response in each voxel of the lane of travel. list-mode data was generated using the same method as described in Section 4.1. To create the combined background and source list-mode data set, background and simulated source list-mode data sets were combined and sorted by time.

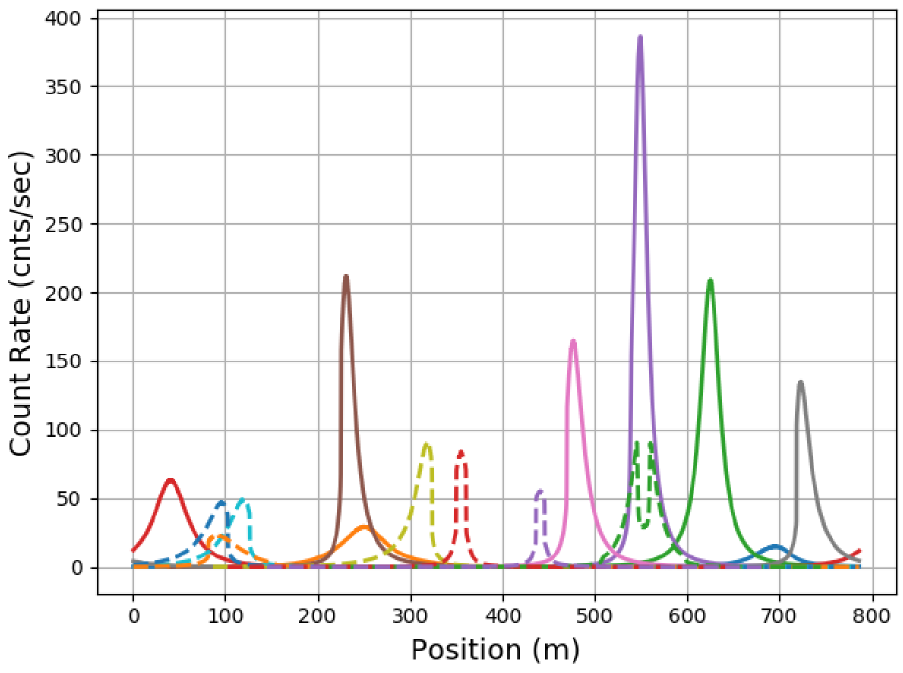


Figure 17: Total count rates for all source locations for a detector in lane 1 of Chameleon Street instance 1 for an IAEA significant quantity of HEU. The position increases while navigating from South to North.

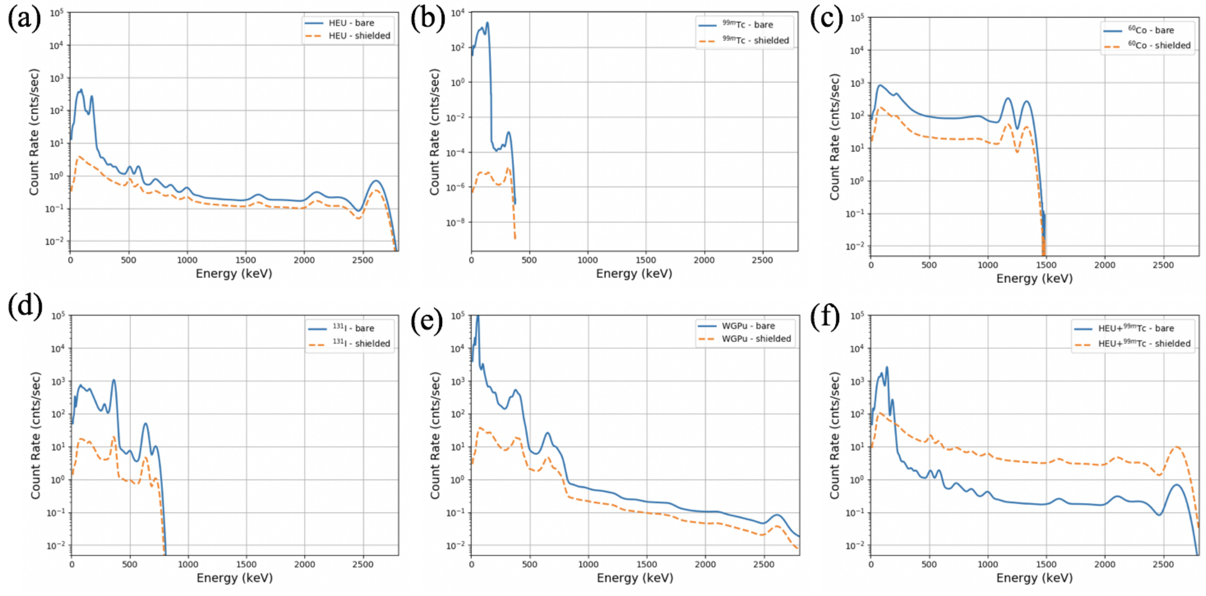


Figure 18: (Color Online) Bare and shielded gamma-ray spectra for (a) 25 kg HEU, (b) 1000 $\mu$Ci $^{99m}$Tc, (c) 1000 $\mu$Ci $^{60}$Co, (d) 1000 $\mu$Ci $^{131}$I, (e) 8 kg WGPu, and (f) 1000 $\mu$Ci $^{99m}$Tc + 25 kg HEU response term scaled to SNR=5.0 at the point of closest approach. All spectra are averaged over lane 1 in background model 1 and energy bins are 2 keV wide.

## 4.3 Parameter Generation

In order to effectively explore a large number of scenarios, the team identified the factors with their ranges of interest that the competition would consider. After discussions, it was determined that 5 sources would be considered, and a 6th source would be created that was a combination of HEU and technetium (see Table 6 for more details). The goal of including the combined source was to assess how easily the HEU could be obfuscated with a more benign source. The amount of source used for each run could be varied, and the tracking of this amount was characterized by a signal-to-noise-ratio that indicated the relative size of the spectral values relative to the ambient background radiation levels. The source could be either exposed or shielded, and there were 15 locations throughout the 8 blocks of the simulated street that could be used for placement of each of the sources.

Table 6: Classification targets for the competition.

| ID | source |
|---|---|
| 0 | null/background |
| 1 | HEU |
| 2 | WGPu |
| 3 | $^{131}$I |
| 4 | $^{60}$Co |
| 5 | $^{99m}$Tc |
| 6 | $^{99m}$Tc & HEU |

A wide range of backgrounds could be obtained by (a) changing the ordering of the blocks in the MUSE simulation, (b) altering the beginning and ending points for each run, and (c) changing the K, U, Th values for each of the material classes in the model. The ranges for K, U and Th were determined based on empirical values that had been observed in different regions across the country.

The detector could move at a speed that ranged from slow walking speed (1 m/s) to moderate driving speed (13 m/s). In addition, the detector could be simulated to be driving in either direction up or down the street, with a choice of 2 lanes in each direction.

In all, there were parameters that could be varied for each of the runs. The source signal to noise ratio, whether it was shielded and the speed of the detector were considered of primary interest. Which of the different street configurations, the K, U, Th parameters, the choice of location for the source, and the direction and lane of travel for the detector were all tracked during the initial analysis, but were found to be non-significant indicators of performance.

The starting and ending points for the runs were considered nuisance parameters, but helped to provide variety for the location of the sources.

However, the ability to change all of these components collectively proved important to give the competitors a variety of looks across the runs that simulated a rich set of scenarios.

To generate the final training, public and private competition data sets, each of the sources and the no source data sets were constructed separately, and then only combined in the final stage. For each source, an initial superset of runs was generated that was approximately 10 times the size of what would ultimately be presented to the competitors. Using JMP software, uniform space filling designs were generated for each background configuration based on the 100+ model parameters. This process was repeated for each of the six sources. For the no source runs, there were fewer parameters to be specified, since the amount of the source, whether it was shielded or not, and the location of placement were not required.

Generating designs of this size was a substantial challenge for traditional statistical software. Our approach was to use composite designs that crossed multiple smaller designs based on subsets of the model parameters. In addition, three replicates of each scenario were generated, and then separate starting and ending points for the run were created to alter their appearance to the competitor.

A baseline algorithm for detection and identification was run on each of the generated runs and this was used to model the probability of correct detection and identification for each source as a function of background, signal to noise ratio, amount of source, speed of detector and lane of travel. Using the predicted probability of success from these models, the level of difficulty for each run was assessed. Using this information, it was possible to sample an appropriate balance of level of difficulty for each of the sources for the respective training, public and private test sets.

The goals in constructing the data set were

1. to have sufficient numbers of runs across the ranges of the input space to be able to estimate performance for each competitor for each source,
2. to have sufficient runs in the no source category to be able to assess the false positive rate,
3. to target the appropriate level of difficulty to distinguish between the competitors (if too large a fraction of the runs were too easy or too hard, then there would be no differences in performance to explore),
4. to have approximate balance between the sources to have comparable precision in the predictions for all sources, and
5. to not have such rigid structure in the numbers of runs for each of the sources that this could be exploited by the competitors.

Hence the competition data set construction was actually based on seven very large supersets (one for each source and one for the no source case). From each of these supersets, six designs were constructed - a training, public and private test set for with and without shielding. The size of each of these designs was chosen to balance

1. the overall size of the competition data set,
2. the goal of having sufficient data to estimate the planned post-competition data modeling,
3. having adequate data for both the public and private leaderboard scoring, and
4. approximate, but not perfect, balance between sources.

Overall, the design construction posed some interesting challenges. The sheer size of the designs, the number of different scenarios to be explored across the cases of interest and the assessment of anticipated algorithm performance were all important considerations to presenting a worthy data set to the competitors. The bookkeeping for these designs and their final combination into the training, public and private test sets required detailed tracking of the components and careful randomization of the runs in the final stages to avoid introducing patterns in the data which could be exploited by the competitors.

## 4.4 Hosting the Government-only Data Competition

Prior to the culminating open data competition (Sec 5), the project team first created a government-only 'restricted' data competition for the purposes of:

- Testing the list-mode data;
- Testing the scoring algorithms;
- Validating the concept of the radiological data competition; and
- Developing an understanding of the efficacy of existing government algorithms against the simulated data set.

Participation in this competition was limited to government employees and others working on federally supported research. By restricting the participation, the project team was able to enforce that the government-only competition did not directly compete with the subsequent open competition and participants were subjected to a usage agreement wherein they were prohibited from disseminating or sharing the competition data and their methods until after the completion of the open competition. In support of the governement only competition, the project team created a website, `https://datacompetitions.lbl.gov/`, hosted by LBNL that provided the following functionality:

- Landing page for user registration that lists existing competitions;
- Support for user registration, the formation of 'team's' and user-based administration of their teams;
- Competition-specific information pages to supply users with instructions, competition concepts, and the ability download the competition data and submission template;
- An upload page and supported back-end scoring algorithm;
- A leaderboard indicating the public scores of each submission; and
- Multiple competition states, active, complete, and 'zombie'.

Those aspects that deserve further detail are described below.

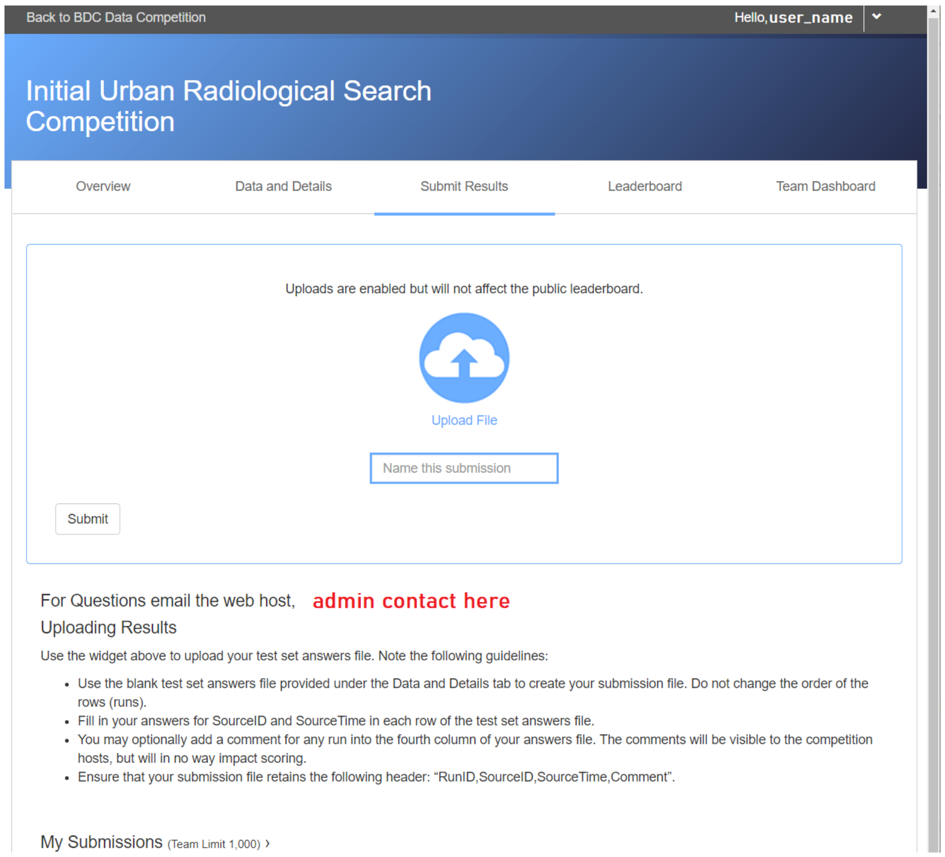
The user registration model was one where individuals could register and either request their own 'team' or to be admitted into an existing team, but would need approval by an LBNL site administrator prior to account activation. The motivation for enforcing manual approval was two-fold. First, the administrator could confirm that the registrant was indeed affiliated with a US government institution, and could politely refer those who were not to the forthcoming open competition. The second motivation was because manual account affirmation is the simplest way by which to conform to LBNL cybersecurity policies. Upon account activation, logged-in users were granted access to download the competition data files, to submit their own answer set, and to view the leaderboard and their team's submission board.

Approval to create or join a team was granted through a separate manual process. The administer always registrants to create their own team, but registrants requesting to join an existing team were subject to the approval of the team. The creation of teams within the data competition environment was one feature that was substantially different between the government competition and the open competition. It also produced some unanticipated benefits. Particularly, that of building collaborative teams within institutions. For example, registrants would often create team names that identified their institution. Subsequent registrants would see that others from their institution had registered and request their team. Since teams requests are not automatically approved, at a minimum this required the new registrant to gain approval from the team's creator and therefore to initiate contact and potentially realize there are shared research interests albeit occasionally very disparate skill sets. Members of some teams formulated in this fashion expressed gratitude that the competition enabled them to meet their peers and formulate professional working relationships.

The competition-specific information was generated to be publicly released. It is ref [92] and the process of creating the data and associated documentation is best described in ref [93]. The competition-specific information was visible to those visiting the competition website without authentication. However, the ability to download the competition data and associated files was restricted to authenticated users. This authentication requirement facilitated restricted dissemination of the competition data, which helped maintain the 'fairness' of the subsequent open competition.

The upload page was designed to be as simple as possible. A screen shot of the page is shown as Figure 19. The page featured drag-and-drop or file-selection based uploading and a field to name the selected upload to facilitate organization of a user's submissions. Upon submission of a solution file the file uploads and is immediately parsed and scored. The combined actions of uploading and scoring typically took a few seconds to run, although uploading is connection-speed dependent. Details on the scoring algorithm are provided in Section 4.4.1. Once a submission is scored, a pop-up window would inform the user the public score of the submission, or if the file had formatting errors the user would be provided an error message and the submission would not be scored. The leaderboard displayed each team's name, best public score, and the number of submissions each team has provided. An additional team Team Dashboard listed all submissions associated with that team. The number of submissions each team was allowed was limited to 1,000. The motivation for this limitation was to reduce the competitors' ability to 'game' the competition by iteratively submitting answers to discern characteristics of the competition that were not intended to be conveyed, such as which individual runs were part of the private data set or exactly how many runs with each different type of source were in the public data set, which presumably could help users infer how many of each type of source they should answer in the private set (although these numbers were somewhat randomized, see Section 6.2).

The government competition began January 22 and ran for 112 days, ending on May 14, 2018. In total there were 66 users, 25 of whom submitted scored results. The users formed 25 teams (5 users never requested



Initial Urban Radiological Search Competition

Uploads are enabled but will not affect the public leaderboard.

For Questions email the web host, admin contact here

Uploading Results

Use the widget above to upload your test set answers file. Note the following guidelines:

- Use the blank test set answers file provided under the Data and Details tab to create your submission file. Do not change the order of the rows (runs).
- Fill in your answers for SourceID and SourceTime in each row of the test set answers file.
- You may optionally add a comment for any run into the fourth column of your answers file. The comments will be visible to the competition hosts, but will in no way impact scoring.
- Ensure that your submission file retains the following header: "RunID,SourceID,SourceTime,Comment".

My Submissions (Team Limit 1,000) ›

Figure 19: View of upload page of the government-only competition website, `https://datacompetitions.lbl.gov/`. Specific user names and website host contact information have been removed from this graphic.

to join a team) and 16 of the teams submitted results. The initially announced end date was April 9 (day 77), but the end-date was twice delayed to allow the participation incentives that DNN R&D program managers provided to take effect. On March 16 (day 53), a revised end date of April 30 was announced (day 98) and then on March 29 (day 66) the final end date of May 14 (day 112) was announced. The cumulative number of submissions throughout the Competition is shown in Figure 20. The original and temporarily revised end-dates are shown in green and the dates on which the end-date revisions were announced are shown in black. Day 66 represented a large up-tick in activity, which waned some until the final week when submissions increased substantially. The final day of the competition had 267 submissions from 15 of the competitors.

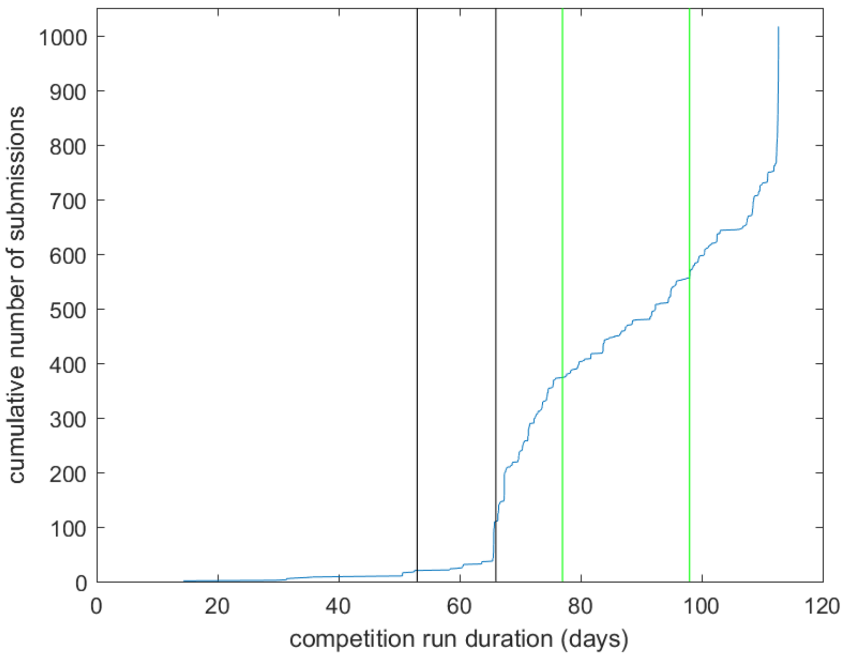


Figure 20: Number of submissions versus duration of the government competition. Vertical black and green lines indicate the dates when the announced end-date of the competition was revised (black) and the two defunct end dates (green).

At the completion of the competition, the website's state was frozen (marking the state as 'complete') and the leaderboard was updated to show the private scores of each teams' best submission.

After the competition, the project team solicited feedback from the competitors. Most of the feedback was quite positive, however, the spectral variability of the competition data set was noted to be lower than is typically encountered in realistic operations. This particular feedback resulted in changes to the data provided in the open competition. Changes in the competition dataset between the government and open competitions included:

- Additional spectral variation in the background environment to further test the competitors and increase the realism of the background found in the real world.
- A shift in the ranges of the SNR for some of the sources to increase the level of difficulty for detection and identification. Ranges of the SNR for which almost all of the competitors got the correct answer were not beneficial for distinguishing between algorithms.
- The relative difficulty of the training, public test and private test sets was adjusted to include more overlap between the datasets. This allowed for easier assessment of and comparison between the algorithm's performance for the public and private test sets.

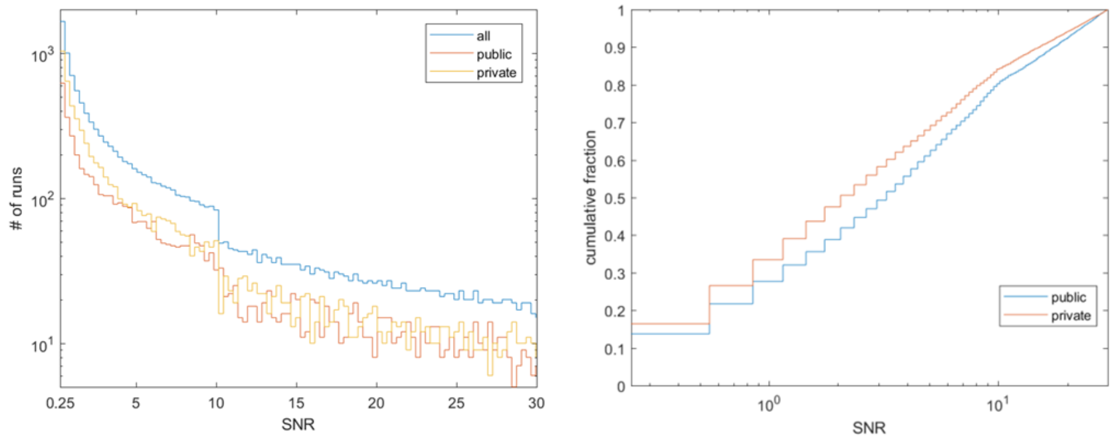


Figure 21: (left) The distribution of the signal-to-noise ratios for the with-source runs in the government competition relative for the public and private data sets. (Right) the cumulative distributions of the SNRs, which better highlights the additional low-SNR data in the private data set.

- The proportion of runs from most challenging source location in the simulated data was increased to be able to assess a highly challenging scenario.
- Since the final winning algorithms in the government competition had false positive rates that were operationally too high, the leaderboard scoring formula was adjusted to increase the penalty on false positives for runs with no source. The implementation of this change directly led to a greater emphasis by the competitors on keeping the false positive rate lower.

Users also noted that the restriction to six sources was unrealistic, as were the decision to separate the competition data set into thousands of individual 'runs', rather than operationally-relevant continuous data collections and the absence of detector effects such as gain drift. Some competitors had asked that we provide feedback on which aspect of the scoring was most adversely impacting their submissions (false positives, false negatives, misidentification or timing). In addition, some users requested an application programming interface (API) to enable them to script the submission of their answers. This feature, while potentially beneficial for the users to accelerate their improvements, was never implemented.

Lastly, several users asked that the government competition website could be made available again so that they could continue testing their algorithms. This resulted in the website being reactivated in August, 2018 in a 'zombie' state (animated, but not live) wherein users could continue to submit and receive scores based on the public portion of the data set, but the leaderboard would no longer be updated.

### 4.4.1 Scoring Algorithm

A key to driving results and improvements by the competitors is to have a carefully developed and tested scoring metric. By its nature the leaderboard scoring need to be defined before the start of the competition, and also combines the numerous aspects that define a good radiation detection algorithm. In both the government and TopCoder the competitions, there were 6 sources that were included in the data, and for each of them, it was important to be able to evaluate the algorithms ability to detect, identify and locate the source. In addition, a good algorithm needed to have a low false positive rate for runs where no source was present. The relative importance of each of these criteria needs to be balanced through the number of runs for each of the sources, and the relative weight assigned to the different characteristics.

In keeping with the standards used by TopCoder, the scores for both the public and private leaderboards were constructed to have ranges from 0 (worst) to 100 (best) and reflect the aggregate success for three components: detection, identification, and location. Meanwhile, the government competition scoring ranged from 0 (worst) to 10,000 (best) on the public leaderboard and the same point scaling, yet differing numbers of runs in the private data set resulted in a scoring range from -2,002.64 to 11,562.11.

Let $B$ be the base score and $p$ be the unit of possible penalty (arbitrary scale). The score is initially set to $B$ points and then it is modified according to these rules:

- For each run that contains a source:
  - Detection: If you incorrectly say there is no source present (false negative [FN]), you lose $2p$ points.
  - Identification and location: If you correctly say there is a source present and the distance $D$ between the predicted location and the correct location is less than the standoff, $b$, which is different for each run:
    - Identification: If you correctly identify the SourceID, you earn $p$ points.
    - Location: You earn points according to how far away you are from the correct location, up to a maximum of $p$ points, following this formula: $p \cdot \cos((\frac{\pi}{2}) \cdot (\frac{D}{\mathrm{b}}))$, where $D$ is the distance between the reported position of closest approach and the actual position and $b$ is the distance of closest approach to the source - a run-specific parameter.
  - Location: If you correctly say there is a source present but get the location wrong (i.e., not within a given standoff), you lose 2p points (referred to as false location [FL]) so it is weighted the same as false negative.
- For each run that does not contain a source:
  - Detection: If you incorrectly say there is a source present (false positive [FP]), you lose 2p points. Otherwise (true negative [TN]), you earn 6p points. For the government competition, TN would only result in 0p points.

The values of $B$ and $p$ are calculated so that the minimum possible score is 0 and the maximum possible score is 100.

Since it is preferential for a working detector not to have too many false positives, successfully identifying runs containing no sources (true negative [TN]) contributes more to the final score than runs with a source. This was changed from government competition, where it was found that winning algorithms were tuned to produce too-high FP rates to be operationally relevant.

## 4.5 A Benchmark Data Set

The data from the TopCoder data competition has been packaged up as a compressed (gzip) tarball (13 GB) and is now available for download [94]. To download the data, users must use either the Globus Connect Personal application to create a personal endpoint, see Ref. [95], or a Globus endpoint server to download the data. Once downloaded, the tarball contains:

- README.pdf–a pdf document with an overview of the data set,
- scorer/–a directory containing a code to score training and test answers,
- sourceInfo/–a directory containing threat source templates,
- submittedAnswers.csv–a template solution .csv file for the test set,
- testing/–a directory containing all test data sets,
- training/–a directory containing all training data sets, and
- trainingAnswers.csv–a template solution .csv file for the test set.

The sourceInfo directory contains source templates for bare and shielded threat sources in SourceData.csv. Plots of all source templates are also included in .png format.

The training (testing) directory contains ASCII and comma delimited (CSV) format training (test) data in list-mode format. The first column represents the time between detected photons in units of microseconds. The second column is the energy of the detected photon in units of kiloelectron volts (keV). For the first event in each run file, the time since the last event is recorded as 0.

Example run file:

```
0,69.0

985,154.7

757,55.5

908,1106.9

1105,356.1

391,116.9

. . .
```

In this example, the first row records the first event, which had a deposited energy of 69 keV. The second row describes the second event, which occurred 985 $\mu$s after the first with a deposited energy of 154.7 keV. The third row records an event that occurred 757 $\mu$s after the second event (985 + 757 = 1742 $\mu$s since the first event of the run) with a deposited energy of 55.5 keV. The event descriptions continue row-by-row until the end of the run. The total number of rows for each file varies.

The scorer directory contains a Python 3 code to generate a score based on the solution templates (solution_training.csv and solution_testing.csv in the scorer/ directory). This code uses training and test set answers keys (answerKey_testing.csv and answerKey_training.csv) for generating training and test scores. A public score is given for both data sets, on approximately 42% of the runs in the test set and 100% of the training set. For only the test set, a private score is generated on the remaining 58% of the runs. When grading the training set, the private score output can be ignored. This segmentation of the test set into a public and private score was done for the purpose of judging the competition. A template answer sheets are available in the /scorer directory along with a README text file describing how to run the scoring algorithm.

Before, during, and after the competition, there was a lot of interest in using this data set for training and evaluating radiation detection algorithms. The data has been used by analytic teams in the MINOS program, other NNSA lab research[72], a Ph.D thesis to develop and evaluate a neural network based radiation detection algorithm [69, 60], and other student projects.

# 5 Mechanics of Holding an Open Data Competition

Hosting a data competition can be highly rewarding for the improvements that can be achieved by leveraging the expertise of the broader community for new algorithms and approaches. However, the process of hosting the competition can be time- and labor-consuming. Hence, it is important that a clear strategy be used to present the competitors with the most advantageous sets of data on which they can develop, train and test their approaches.

In this section we outline the key steps in the development and fielding of the data competition.

1. Identify clear and measurable goals for the competition
2. Assess adequacy and sufficiency of data
3. Determine inputs to manipulate and their ranges of interest
4. Create a superset of all potential data
5. Identify data of maximum interest
6. Construct training, public and private test sets
7. Construct leaderboard scoring
8. Beta test competition
9. Launch and host competition
10. Conclude competition and announce winners
11. Perform detailed analysis of results using a post-competition analysis.

A critical part of the success of the competition is to have clear objectives for what a desirable participant solution should be able to do, and ensuring that the available data are adequate to match the goals. Precisely identifying factors to be explored and their appropriate ranges will frame the space over which answers are sought. We suggest, where possible, to have available a set of candidate data that is considerably larger than the training and test sets that the competitors will ultimately receive. Instances with desirable properties can be selected from this "superset" to create an ideal set for each element of the competition. Desirable instance properties include being in regions where the solution is not too easy or too hard (where differences between solutions are more likely to show), and that the training, public test, and private test sets provide a progression of challenges that encourage solutions to demonstrate their ability to interpolate and extrapolate to avoid overfitting.

Since the leaderboard is typically driven by a single number summary of good performance, selecting how to balance multiple objectives with the appropriate emphasis on each component is important to ensure that the winning competitor produces the overall best solution to the problem posed. Since there is considerable complexity to this competition set-up, we recommend some thorough beta testing by people not involved in the development. This can help ensure that the fielded competition does not have any unintended problems.

When hosting the competition, using a service provider, such as TopCoder or Kaggle, can help to ensure that logistical issues are handled by experts. It is important that the sharing of the data sets, the real-time scoring of the leaderboard scoring for competitors, and the enforcement of rules are all implemented and executed well.

Finally, when the competition is completed, it is important to maximize the investment in the competition by conducting detailed post-competition analyses to extract comprehensive understanding about the relative strengths and weaknesses of the algorithms, differences between the approaches used, and how these characteristics of performance and approach are connected.

## 5.1 Creating the Data

We illustrate the details of the steps outlined above with description of how we implemented the TopCoder competition for the urban radiation detection problem. In that competition we used simulated data to find effective algorithms to detect, identify, and locate a variety of radiological sources from the measurements collected by a radiation detector as it's driven along typical city streets. While the overall goal is to evaluate each of these three objectives separately, we also need to formulate a single-number value by which to judge performance on the leaderboard. A post-competition analysis will consider the objectives individually to gain understanding about the strengths and weaknesses of the different competitor approaches.

Several aspects of the input space were manipulated to mimic the breadth of urban environments seen in practice. These were divided into three broad categories: background, sources, and detector movement. For the background, multiple versions of urban streets were created with different building and feature configurations and compositions; see Figure 2 for an example. Five different radioactive source types were included, with an additional source being defined as a combination of two of the sources. The locations on the street, the strength of the source, and whether it was shielded in a dampening container were also varied. A key feature of urban detection is being able to separate the background signal (generated from the urban environment) from the local source. Finally the movement of the detector was varied, by changing its speed, its traffic lane, and its starting and ending points.

Each instance or "run" of data required setting more than 100 parameters, and individual file sizes ranged from 160 KB (when the detector is moving quickly and over a shorter section of road) to 7.3 MB (moving slowly over a longer path with more active background). The target total file size for the zipped data was 10 GB, which constrained the number of scenarios which could be shared with the competitors in the combined training and test data sets. After the initial development process for the different scenarios, generating the superset of data (a collection of 100GB of candidate runs to be considered) took under a day of computer time on a large computer cluster.

As noted in the traditional design of experiment (DoE) literature, matching the design region to the study problem of interest is essential for being able to answer the right question. This entails identifying the factors to be varied, ranges of each of the factors, and potential constraints on viable factor combinations that may make the region to be explored irregular. As with traditional DoE, subject matter experts (SMEs) typically consider a larger number of candidate factors, and then down-select to identify those that are thought to be most influential. This changes for a data competition because of constraints on what data might be available. If the data are being simulated, then the capability of the data generator may restrict which factors are considered and what ranges are available. If a subset is being selected from an available collection of real data, then relevant input factors may not have been measured or some factors may only be available in a portion of the range of interest. The constraints on what is easily available should be balanced with the study goals, and finding creative ways to expand the available set of data can often improve the ability of the competition to actually answer the real aims of the study.

Since the generation of the data (once the simulator was complete) was relatively inexpensive, we generated a "superset" of data that was approximately 10 times the intended final size of the competitor data set (approximately 100 GB of data). For each of the six sources, a design of experiment strategy was used to create a space-filling design using JMP software for each of the different street configurations involving all background and detector movement factors. For the factors with fixed levels, the design was balanced. For continuous factors, the range was partitioned into 5 sub-ranges for generating the space-filling design, and then values within each sub-range were randomly generated from a uniform distribution. The design was then replicated for each street configuration, with different values sampled for the continuous factors. This background design was crossed with a full-factorial design for each of the source factors. This created a large super-design involving over 100 factors, and three replicates of each configuration were generated. Exploration of the diversity of the backgrounds showed that replicates within a set of fixed parameter values were relatively consistent, while across the breadth of the experiment ranges the backgrounds looked very different. This provided reassurance that the goal of having the competitors develop general algorithms to detect in a diverse urban environment was viable.

One of the key elements that we propose as a strategy is that the focus of the competition should not be on obtaining a global estimate of performance across typical data. Instead, we think that the most advantageous way of accelerating the improvement of the solutions is to intentionally vary the mix of data from what is typical. If we wish to trigger solution improvement for challenging aspects of the problem, then sufficient examples of that type must be provided for the competitor to see. With the overall size constraint of the competition data, the most interesting non-trivial scenarios are often too severely underrepresented in the typical data to offer sufficient information for the competitors to train their algorithms. Oversampling the more interesting observations allows increased flexibility about what data to select, but does have the byproduct that the probability of getting the correct answer should only be interpreted locally for a given set of inputs, and not taken as a representative global proportion across typical data.

The possible data to include in the training, and public and private test sets do not all have similarly informative value. For example, there may be some instances where answering the question of interest is trivially simple. In other cases, the instances may be impossibly hard with little practical possibility of having sufficient information to solve the problem posed. The sweet spot for providing data to the competitors is in the middle range, where there is a good possibility of getting the right answer if the algorithm is sufficiently capable. Since a primary goal of the competition is to be able to distinguish between the performance of different competitors, choosing the majority of the data in this middle range is advantageous. Having a lot of data where all of the competitors will get the same answer (all get it right, or all get it wrong) is an inefficient use of resources given the total data file constraint. As a simple illustration, consider a detection problem where the competitors are asked to determine whether a source is present or absent. A logistic model based on the levels of the input factors can be used to model this relationship. If we further simplify and just consider a single input with a known relationship to the probability of correctly detecting, a D-optimal design places half of the points at the location with a probability of success 0.176 and the other half at the location with a probability of success 0.824 [1]. As illustrated in Figure 3, input levels that yield probabilities below the lower value location in the D-optimal design would be too hard for most of the competitors, while input combinations above the higher value location would be too easy. Neither of these extreme regions would help identify the most promising solutions. The middle region instead will provide more informative data for distinguishing competitors.

Hence the goal for creating the data sets should be to provide sufficient data in the anticipated regions of interest to allow for good estimation for each of the competitors near the top of the leaderboard.

For the urban search competition, we used an available detection and identification algorithm on the entire superset of data. The data were partitioned by source (plus one set of data for runs containing no source) and the results from the algorithm to identify each source were used to obtain fitted logistic models for each source that were a function of source strength and shielding, background level, and detector speed. Since identification was the most important of the three goals of the study, we focused primarily on this for defining the most interesting region of the superset. Using this model, regions in the input space were identified that were sufficiently difficult for this algorithm to justify their inclusion in the competitor data sets. In addition, we consulted with subject matter experts to frame the region where they thought that an exceptional algorithm might be able to discern a signal, both for detection and identification. The most difficult region for identification was used as an upper bound. However, to reduce dependence on using this as a prior, and since the range of interest for each of the inputs had been determined separately, we opted to continue to use the entire range for each of the inputs, but to weight the more promising regions more heavily. Hence for each source, regions that were deemed to be "easy" with $P(\text{identification}) > 0.824$ were weighted very lightly, since the current algorithm could already solve this problem. Regions of the input space with $0.5 < P(\text{identification}) < 0.824$ were sampled moderately heavily, since success here was already likely but was not guaranteed with the current baseline algorithm. Regions of the input space with $0.176 < P(\text{identification}) < 0.5$ but still within the region that the experts thought was potentially attainable were sampled heavily since it was hoped that improving algorithms might be able to grow into good performance in this region, and we wished to have good ability to distinguish between competitors here. Finally, the region with $P(\text{identification}) < 0.176$ were sampled moderately heavily, since this region was currently deemed very difficult, but we did not want to preclude being able to identify an extremely good algorithm at the conclusion of the competition.

Once this exercise had been completed, we evaluated the chosen prioritization in the context of the detection goal. Since there were still lots of highly challenging regions for the identification aspect, the subject matter experts thought that the amount of data to evaluate the detection aspect would be adequate for comparisons between algorithms. At the conclusion of this phase, the entire region of the input space was still represented, but some regions were emphasized more heavily than others as a reflection of their anticipated relative importance to understanding and comparing competitor solutions.

One of the obstacles to using data competitions to develop long-term solutions for complex problems is the required static nature of the data sets. In order to have fair comparisons between competitors and for them to understand the requirements of the solution, the data sets remain unchanged throughout the competition. From the competitor perspective, this provides opportunities for learning from their previous submissions and experimenting with adjustments to the solution algorithms. Particularly, if the training and test sets share similar performance characteristics, then it is even easier for competitors to improve the leaderboard score by continuously increasing the model complexity to capture the idiosyncrasies found in the training data. With repeated experimentation, the algorithm can be tuned to maximize the team's leaderboard score. From the host perspective, this fixed set of data could potentially lead to competitors solving the wrong problem – rendering the developed algorithm ineffective when used on a more general solution space. The potential risks of model overfitting based on a single data set are well documented, but this problem is exacerbated because of the repeated submission aspect of data competitions. Hence, it is important to use the construction of the training, public test, and private test sets as a way of mitigating overfitting and encouraging generalization to unexplored scenarios. Two strategies are recommended: 1. Force the competitors to demonstrate their solutions in expanding regions of the input space, and 2. During the analysis phase, monitor changes in performance between the public and private data sets to quantify the effect of overfitting in the presence of feedback from multiple submissions.

To force competitors to handle new scenarios well, it is helpful to construct the training, public test, and private test sets with increasing levels of difficulty. By not providing answers in the training set to the most difficult scenarios that the participants will be asked to compete on, their algorithms will need to be robust to solving new challenges. In addition, since the private test set will ultimately determine the winner, it is helpful to include new scenarios that the competitors could not tune their algorithms to through multiple submissions against the public test set. In this way, the private test set data provides a good proxy for assessing how algorithms might be expected to perform when a new possible scenario is considered.

The reasoning behind these definitions of the different data sets is to force the algorithms to demonstrate their ability to handle interpolation and extrapolation. Based on the training data, competitors will not have seen answers in the more difficult region in the public test set (light blue perimeter outside of white training set), and hence their solutions will need to demonstrate the ability to expand into new scenarios. This process is repeated for the stretch from the public to private test sets, where the hosts can see how well the algorithms again extend to further new scenarios. Ultimately, the host wants the winning solution to perform well, not only in the competition setting, but also in new (perhaps currently unanticipated) scenarios. The interpolation elements (holes) provide an opportunity to check if there are differences in local behavior that can be a symptom of overfitting.

For the urban radiological search competition, the starting point for creating the three data sets was to clarify which levels of each factor correspond to more challenging conditions. First we consider the extrapolation part of specifying the data. Using a currently available detection algorithm, the superset of data throughout the input space was used to estimate a logistic model for detection capability as a function of the inputs. Based on this estimated model, detecting a source becomes more difficult at higher speeds. Similarly, shielding a source makes it more difficult to detect or identify. Some of the background conditions also were estimated to make the search more difficult. For each of these factors, where there was an anticipated gradient from easy to hard, sub-ranges for the training and public test were specified.

## 5.2 Creating the Leaderboard

With the data sets for the competition constructed, one key aspect of the competition still needs to be developed. The leaderboard scoring provides a single formula metric for ranking the competitor solutions from best to worst. If this ranking is not strategically chosen to match the goals of hosting the competition, then several undesirable outcomes are possible: (1) the competitors focus on aspects of the problem that are of lesser importance, and/or (2) the overall winner does not provide the most desired solution to the competition. It is important to remember that competitors will not be driven by solving the right problem, but rather by maximizing their score. Hence the burden of providing the right encouragement to the participants to focus on important aspects of the problem lies with the competition host. Similarly, once the formula for the leaderboard scoring is set, there is typically no opportunity to change it if solutions are migrating in a direction that does not match the host's goals.

One aspect that is helpful to consider is that each instance in the test set does not need to contribute the same amount to the overall score. The contribution of a particular objective to the overall leaderboard score is a function of two elements: how much each instance in that category is worth, and what fraction of the test set is comprised of data from that category. For example, for the urban search competition, there was interest in moderating the false positive rate. This interest is driven by operational settings in which one wants to ensure that responders are not overwhelmed by following up on too many false alarms. Since "no source" is the typical condition of an actual run, a larger fraction of the data was dedicated to "no source" instances, and hence in the scoring algorithm each of these runs contributed half as much to the "with source" scenarios. The "with source" scenarios involve considerably more work by the competitors – detecting that a source is present, identifying which source it is, and locating where along the path it is found. Hence, each of these runs was weighted more, but there were fewer of them in the total test set.

To determine suitable weights for each of the components, a number of answer files, which were intentionally altered from the perfect answer key, were constructed, and different weightings for the three components were applied. Initial subject matter expertise was used to suggest potential weightings that were thought to reflect competition priorities. Mock leaderboards for these different candidate weightings were constructed to rank the constructed answer files. For example, changing the weights changed the position on the leaderboard of a submission with excellent identification but more false positives, relative to a submission with excellent identification but poorer ability to locate the source. Subject matter experts then identified which ranking of the different scenarios best matched their priorities for a winning submission.

As with many data collection strategies, ensuring that the data closely match the goals of the experiment is essential for efficient use of resources. For some competitions, the primary objective is simply to find the best solution for the set of data provided. In other cases, there are multiple goals which need to be simultaneously accomplished. For the urban search competition, the goals included identifying the best solutions in different regions of the input space (not necessarily just the global winner), understanding the characteristics of performance of all of the solutions across the input space, and characterizing the regions of the design space where the problem remains largely unsolved. And these aspects were explored for each of the detect, identify and locate objectives. The data presented to the competitors was intentionally chosen to allow answers to each of these questions.

We emphasize that the results of data competitions should not be interpreted as giving an overall assessment of how well an algorithm or solution might do when implemented. Recall that the choice of which data to include in the competition was based on having good ability to estimate performance in different regions of interest, and this renders a strategy to over-represent the more challenging scenarios and under-represent the easier ones given the size constraint of the overall competition data. For example, in the urban search competition, the number of with source runs was dramatically over-represented (we hope!) relative to every day experience.

## 5.3 Choosing a Website Host and Post-Competition Analysis

The choice of which service provider to select for hosting the competition website was based on their capability to support the rules and process that the host wishes to implement. For this competition, the priorities that we wanted to incorporate included: (1) flexibility on the number of submissions per day, but with an overall cap to the number of submissions, (2) having the competitors actively declare which submission(s) they wanted considered in the final scoring, (3) the ability to have access to the .csv files for all of the submissions from all of the competitors, and (4) the code that the competitors used for their winning submissions would be collected and available for further analysis. The TopCoder provider was able to satisfy all of these requirements.

Providing the prize money to an open competition field using government resources proved to be one of the major challenges of hosting the competition. Several times it looked like a path to allow this had been found, before a new obstacle was introduced. Ultimately the solution to the many logistical challenges that the distribution of prize money introduced was to worked with a NASA group with extensive experiences hosting open competitions. They were able to act as a coordinator for the bid process to solicit RFPs from outside vendors as well as the mechanism for the transfer of funds to TopCoder, who ultimately was in charge of the distribution of prize money to the competition winners.

## 6 Results and Statistical Methodology

In order to implement the competition with the features that were desired, new statistical methodology was developed, and some existing exploratory analysis methods were adapted to summarize what was learned from the competition results. Throughout this section, the results shown are for the private portion of the test set, which is the set used to determine the final ranking of the competitors and is likely most representative of future performance of the algorithm. The final submission for each competitors was used to determine their ranking and all of the analyses in this section are based on these results.

### 6.1 Outcome of the Competition and Algorithm Performance

One beneficial way to explore patterns in the results is to look at the correct identification and detection proportions for each of the 6 sources for the TopCoder top 10 prize winning competitors, as well as several established baselines and 2 non-prize winning competitors [6].

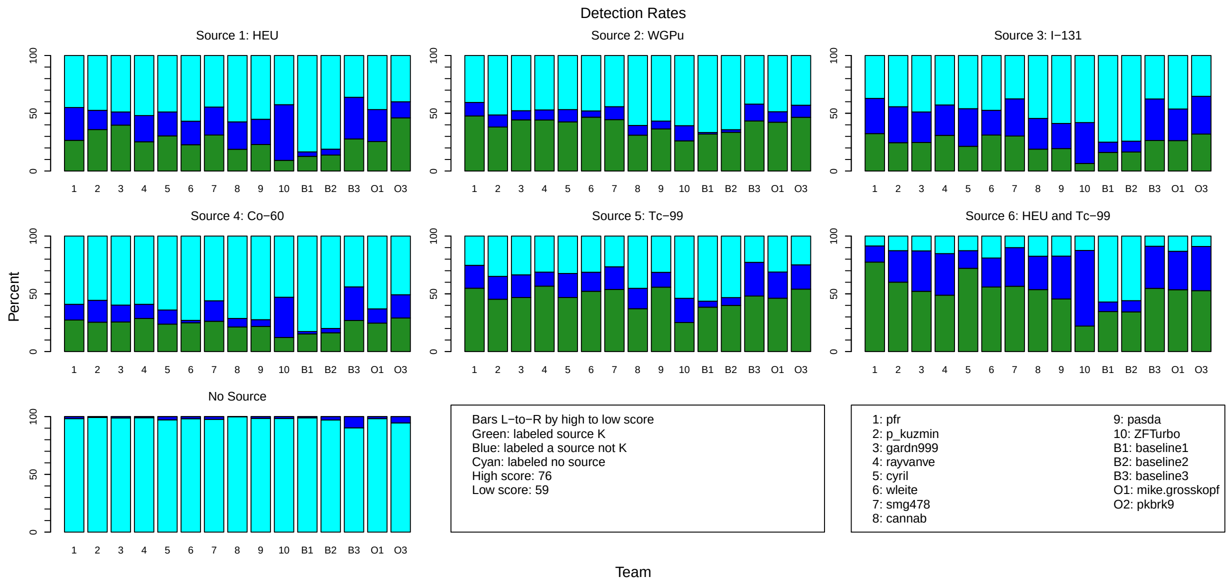


Figure 22: Bargraph of top competitors compared to baselines and two non-prize-winning competitors.

The barplot in Figure 22 shows the results for each source (and the no source subsets of data. The order of the bars is kept consistent throughout all of the subplots, with the first 10 bars corresponding to the prizewinners, in the order that they were ranked by the private leaderboard. The last 5 bars correspond to different baselines that were scored for the leaderboard after the competition was completed.

For the subplots for each source, green indicates that proportion of runs for which the source was correctly identified. The dark blue corresponds to the proportion of runs where the source was detected, but misidentified as one of the other sources. The combined height of the green and dark blue corresponds to the proportion of detected runs for that source. Finally, the light blue corresponds to the proportion of runs where the source was not detected, and the run was declared to have no source.

For the runs with no source, the light blue corresponds to runs that were correctly specified as having no source, while the dark blue corresponds to the proportion of the runs for which the competitor said that a source was present. This fraction corresponds to the false positive rate of the algorithm as measured by the proportion of runs.

One interesting feature of the results is that the order of performance across the different sources is not consistent. For different sources, the best competitor for identification and detection varies among several different prizewinners. The top 10 competitors all have a low false positive rate, as this was something that was penalized harshly with the leaderboard scoring.

### 6.1.1 Confusion Matrices

Next it is possible to consider the identification performance for each of the top 3 competitors compared to the top performing baseline. Figure 23 shows a graphical summary of the confusion matrix for the baseline, WAVRAD for the private test data runs. Each row of the plot represents the data from one of the 7 subsets of data (6 sources, plus the no source subset). Each column represents what the competitors indicated what the result of the run was. For example, if we look at the first row and last column, this indicates what proportion of the runs that actually contained some source 1 (HEU), were scored as No source by the WAVRAD algorithm.

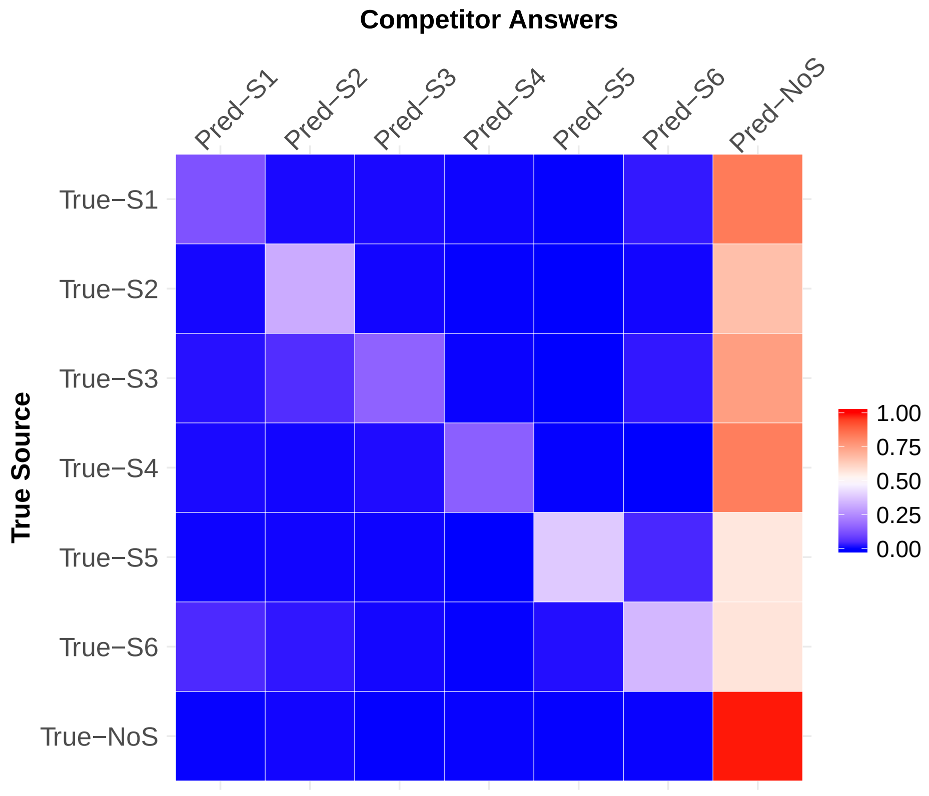


Figure 23: Confusion matrix for baseline, WAVRAD. Rows indicate the true source, while the columns indicate the source indicated by the algorithm.

An ideal plot would have red along the principal diagonal, and dark blue everywhere else. This would indicate that all of the runs for a given source were correctly identified as their correct type, and there was no misclassification.

The results from WAVRAD indicate that many of the runs were not correctly detected or identified, as the largest proportion of the runs for each of the 6 sources were labeled as No Source.

Figures 24 to 26 show the results for the top three competitors. Recall that the winner of the competition had a private test score that was notably higher than the rest of the competitors. The second and third place competitors were behind the winner, but had some separation from the remaining competitors. Hence, for many of the summaries provided, we will focus primarily on the top 3 finishers and examine their results more closely.

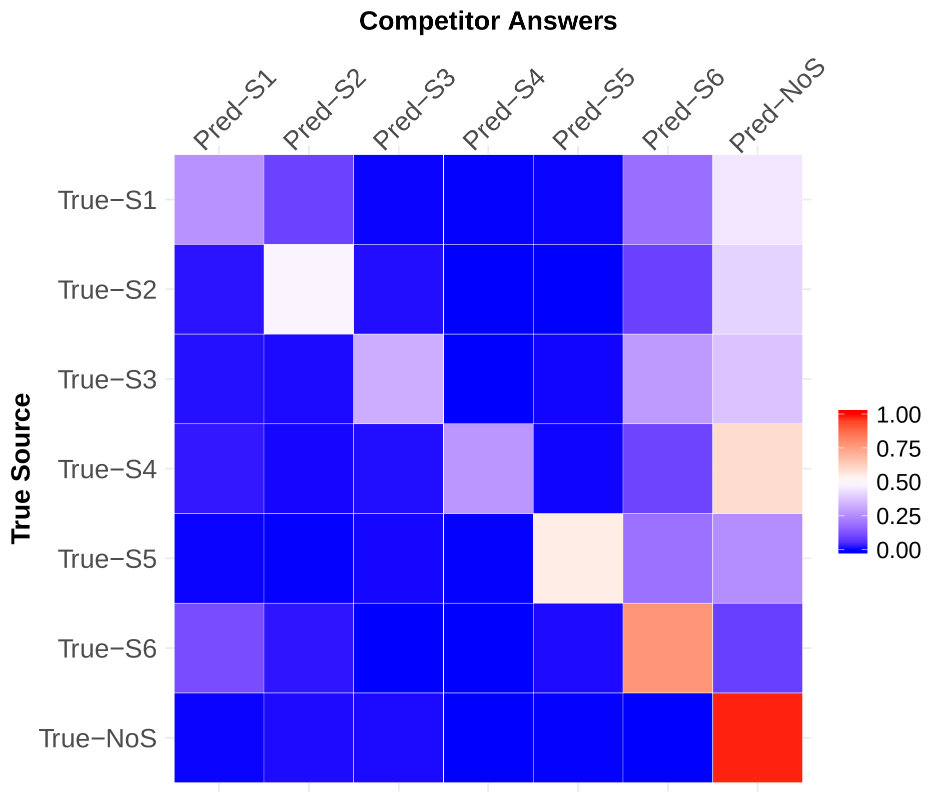


Figure 24: Confusion matrix for TopCoder winning competitor.

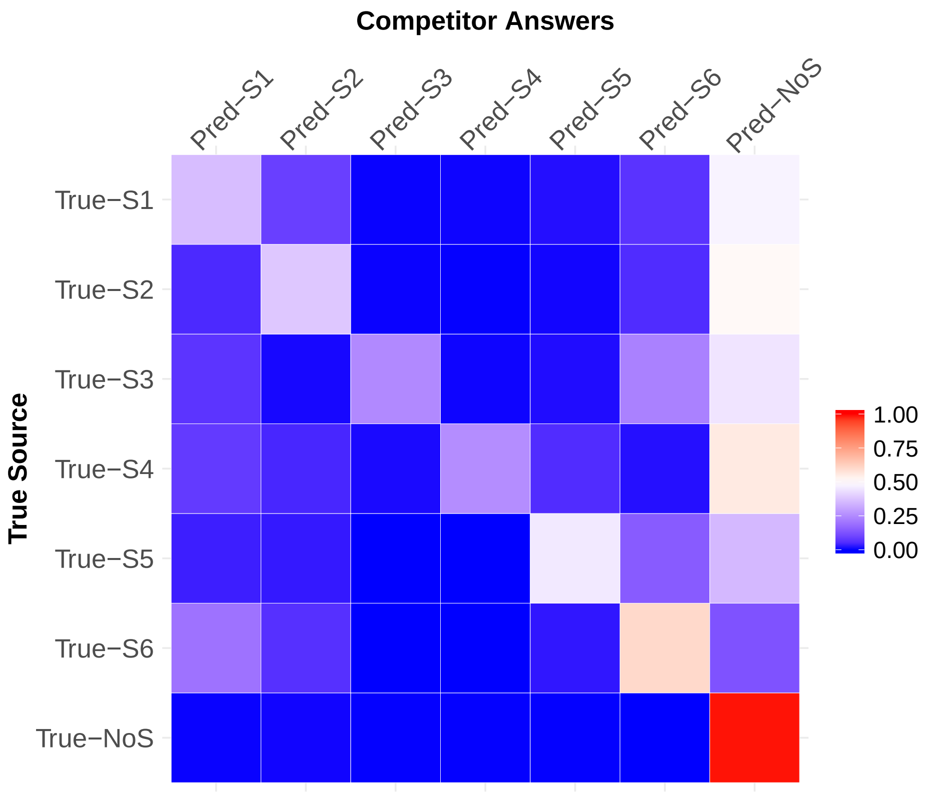


Figure 25: Confusion matrix for TopCoder second place competitor.

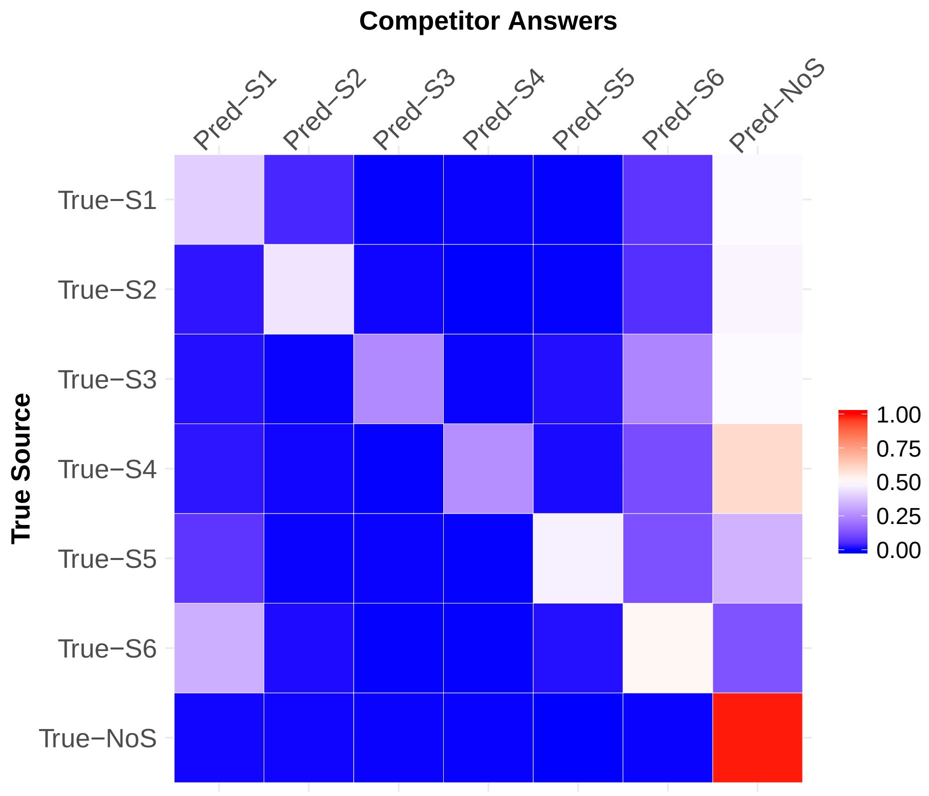


Figure 26: Confusion matrix for TopCoder third place competitor.

When we compare the results of the top three competitors, we see that they have much higher detection rates than the WAVRAD algorithm. This matches what was seen in Figure 22, and in the confusion matrices shows up as the rightmost column having much smaller proportions.

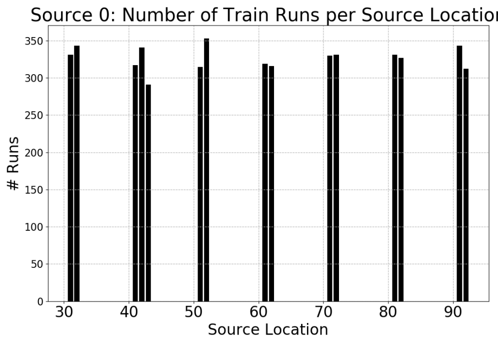
Across all of the top three competitors, the most frequent confusion between the sources was that if a run was misclassified it was most commonly labeled as source 6 (HEU + $^{99m}$Tc).

For sources 1 through 4, the winning competitor has lower false negative rates than the second and third place finishers. There are considerably differences between the patterns of performance for the different sources, and among the competitors.

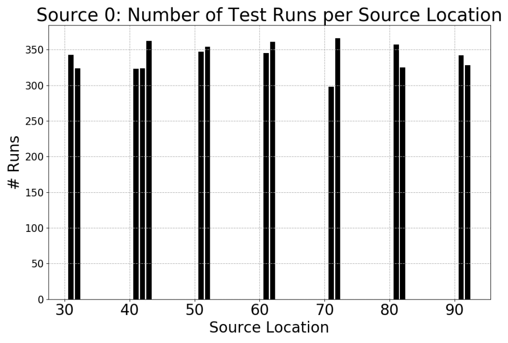
### 6.1.2 Analysis by Source Location

An additional consideration was how much variability there was in the performance of the algorithms at different locations for the source. In the simulated data there were 15 locations along the mapped road where the sources could be placed. These locations were designed to be of varying difficulty with some of the placements of the sources making them partially or fully invisible to the vehicle carrying the detector. Other locations were designed to be more straightforward with the source placed in the open. The location labels range from 31 to 92, where the first digit indicates on which block of the simulated road the source location lies. The second digit indicates which placement within that block the location is. Each of the 7 blocks has two possible locations, with block 4 having three locations. The number of runs in the training and testing sets for each source and each location are plotted in Figures 27 and 28.

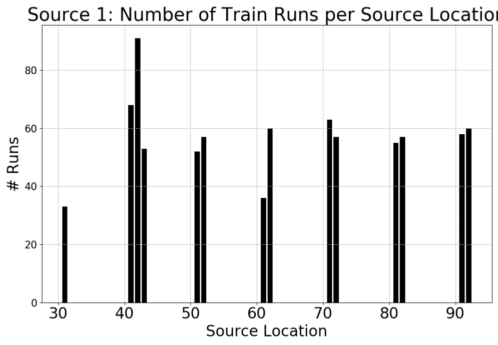
To assess differences in the level of difficulty of the locations, Figures 29 through 34 show the overall proportion of correctly detected and correctly identified runs for each of the sources for the top 3 competitors. The x-axis shows the location on the street. When combined in the proportions that were used in the test data set, these values correspond to the global summaries in Figure 22 for the top three competitors. The black dot corresponds to the first place competitor, while the blue and red dots correspond to the second and third place finishers, respectively.



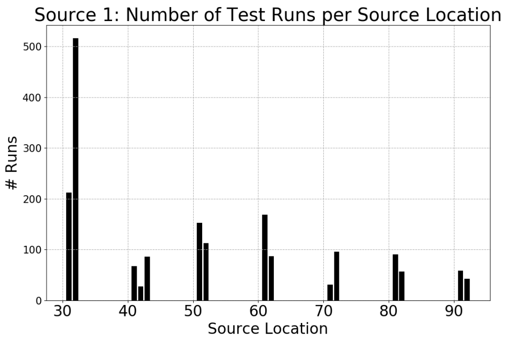
(a) Background :: Training



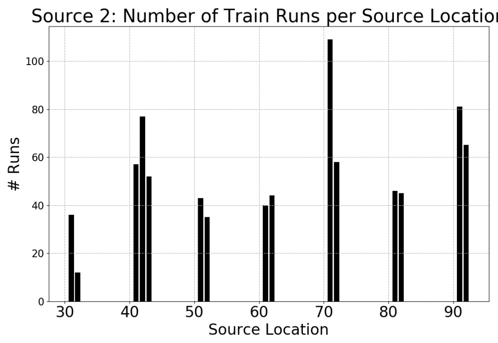
(b) Background :: Testing



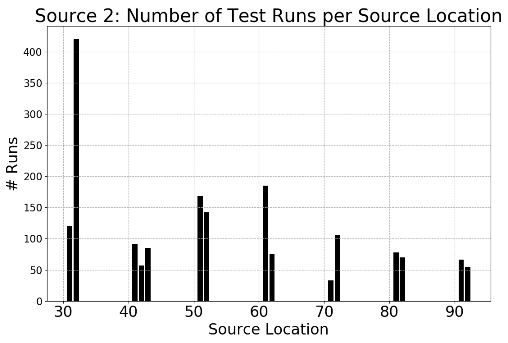
(c) Source 1 :: Training



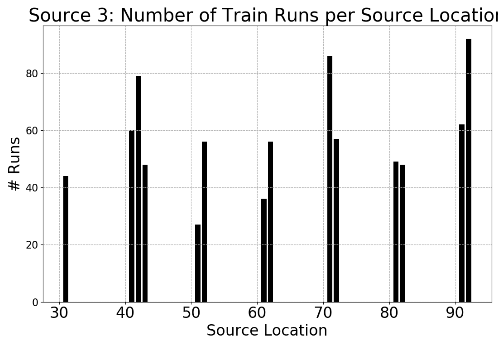
(d) Source 1 :: Testing



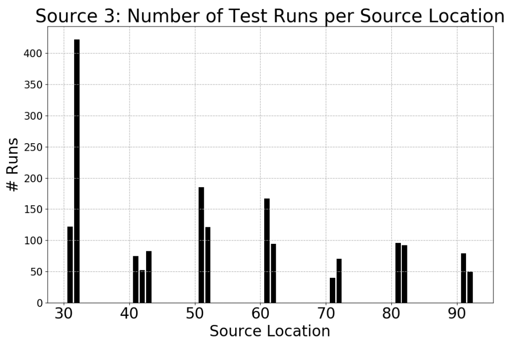
(e) Source 2 :: Training



(f) Source 2 :: Testing

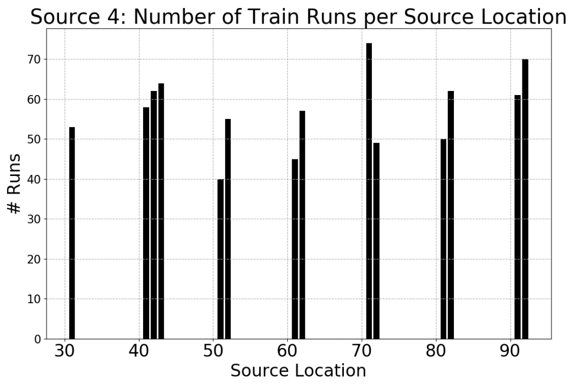


(g) Source 3 :: Training

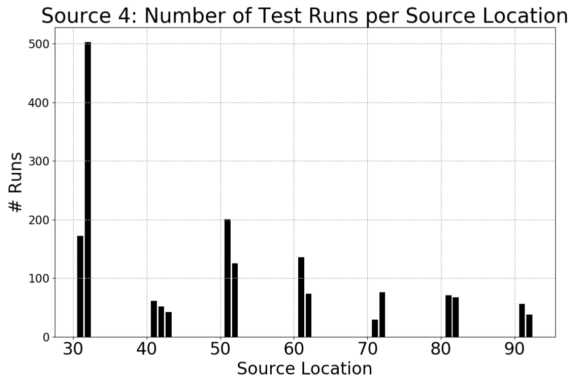


(h) Source 3 :: Testing

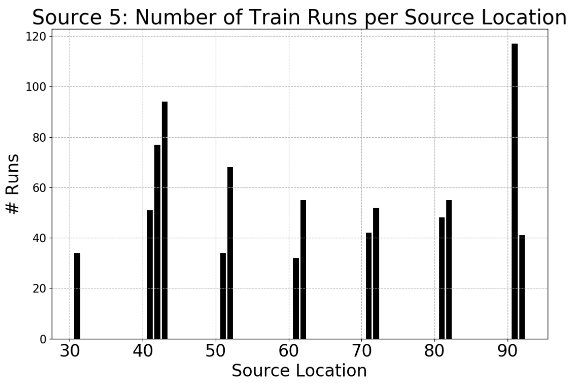
Figure 27: Number of runs in the training (left column) and testing (right column) data sets at each location for background and sources 1, 2, and 3.



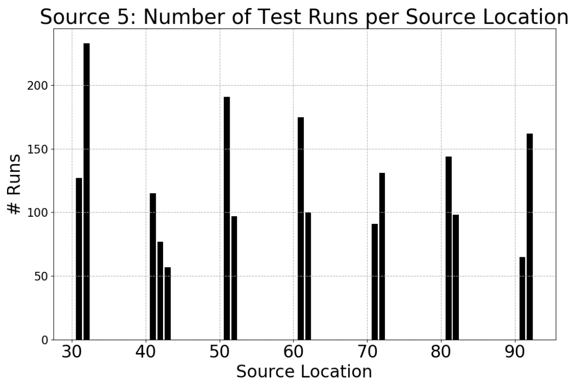
(a) Source 4 :: Training



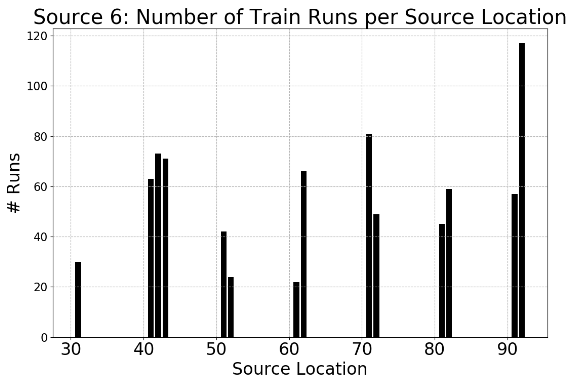
(b) Source 4 :: Testing



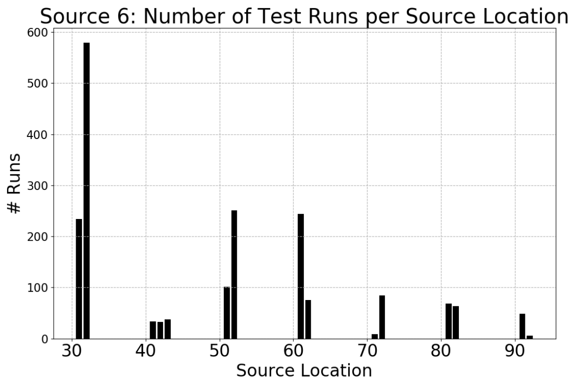
(c) Source 5 :: Training



(d) Source 5 :: Testing

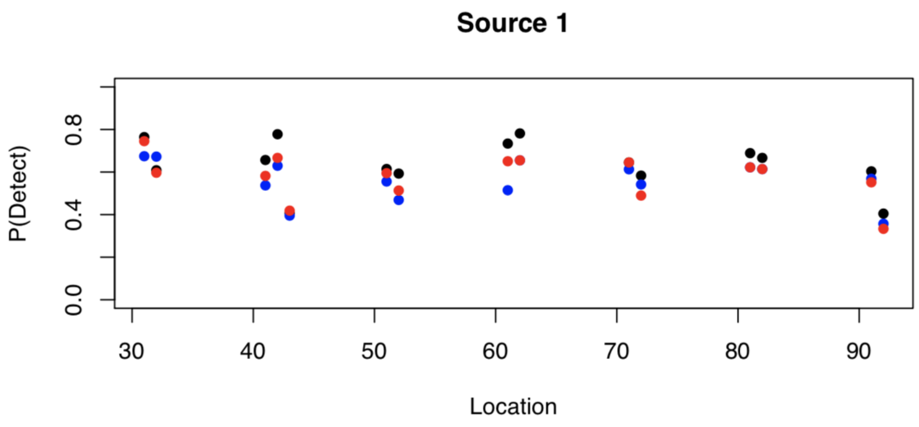


(e) Source 6 :: Training



(f) Source 6 :: Testing

Figure 28: Number of runs in the training (left column) and testing (right column) data sets at each location for sources 4, 5, and 6.



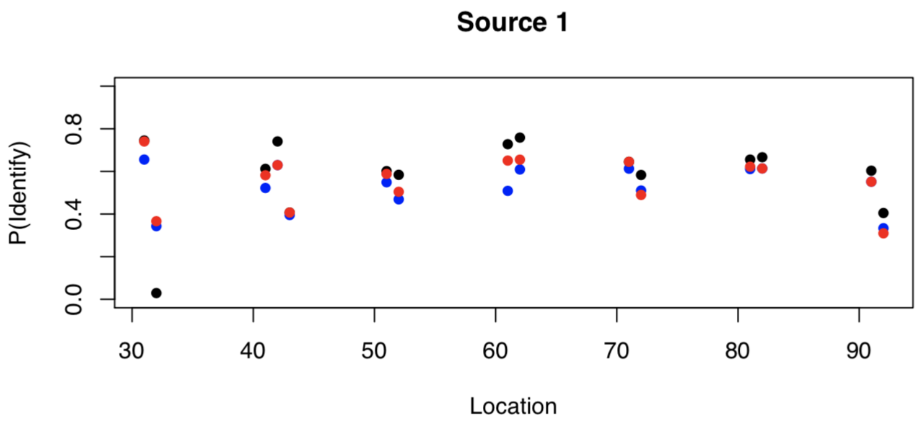
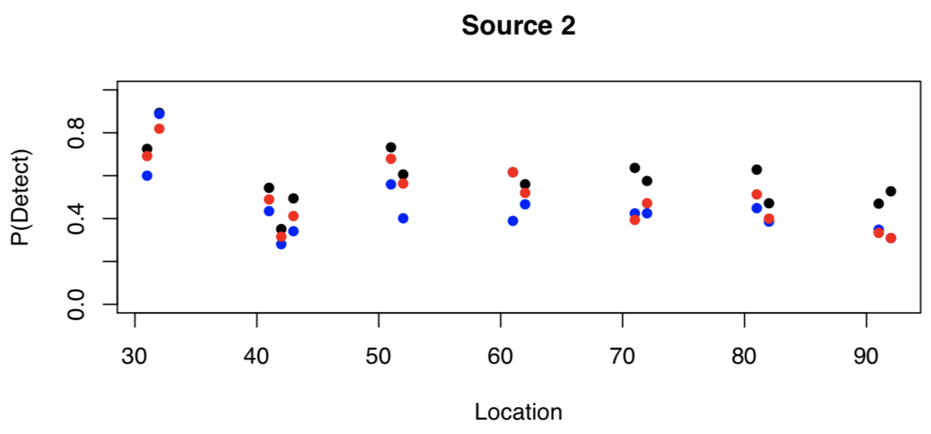


Figure 29: Proportion of correctly detected and identified runs for source 1 (HEU). Black: 1st place, Blue: 2nd place, Red: 3rd place.



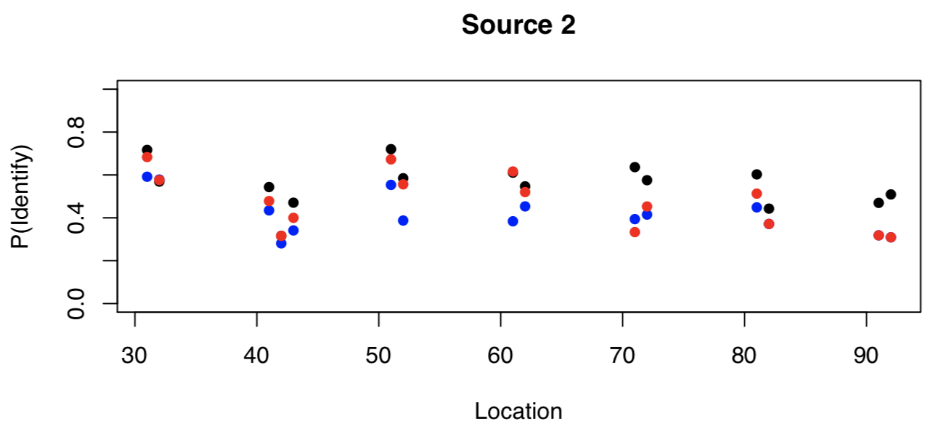
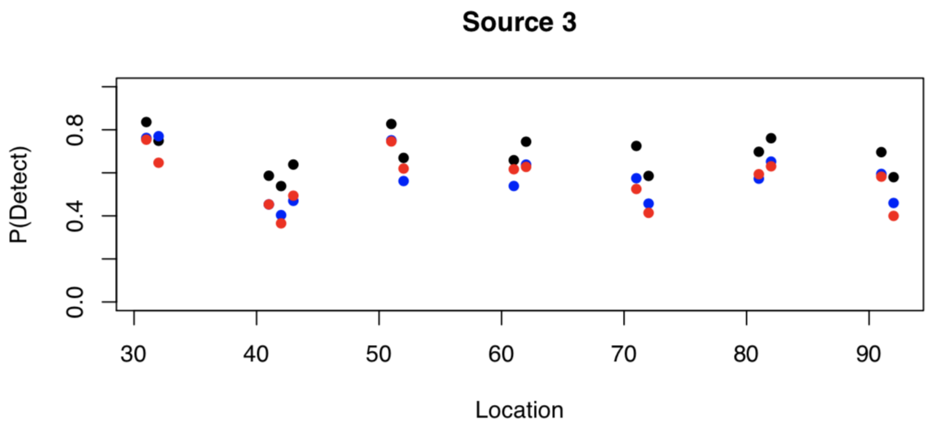


Figure 30: Proportion of correctly detected and identified runs for source 2 (WGPu). Black: 1st place, Blue: 2nd place, Red: 3rd place.



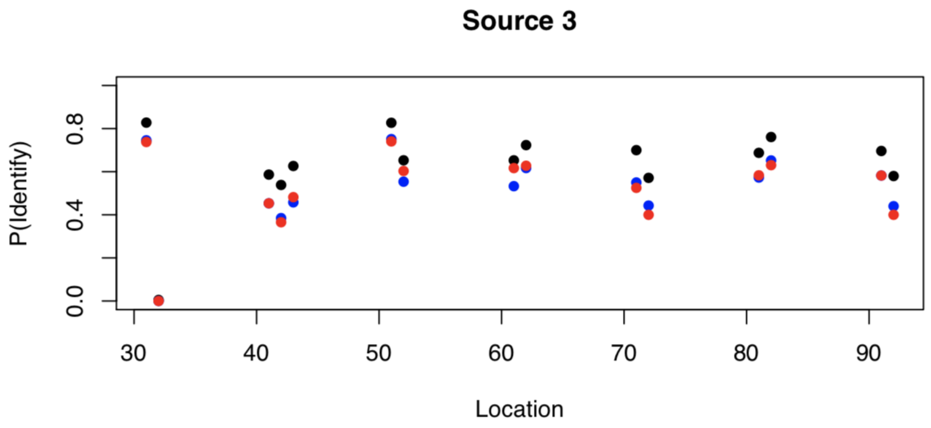
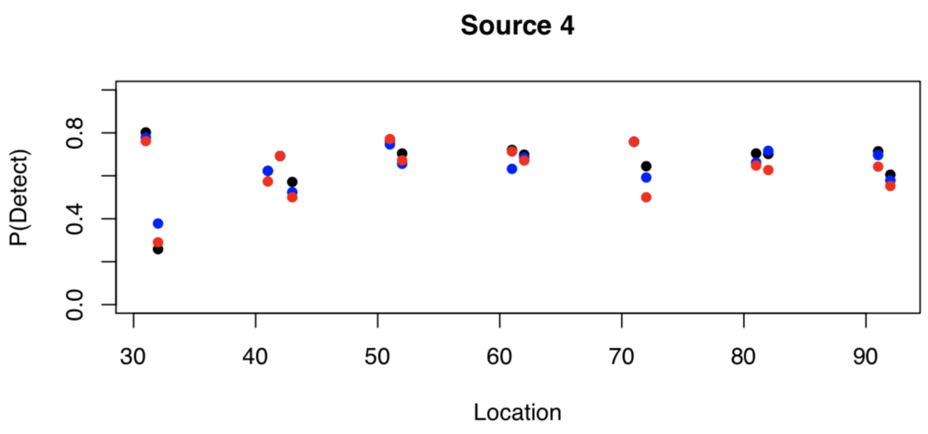


Figure 31: Proportion of correctly detected and identified runs for source 3 ($^{131}$I). Black: 1st place, Blue: 2nd place, Red: 3rd place.



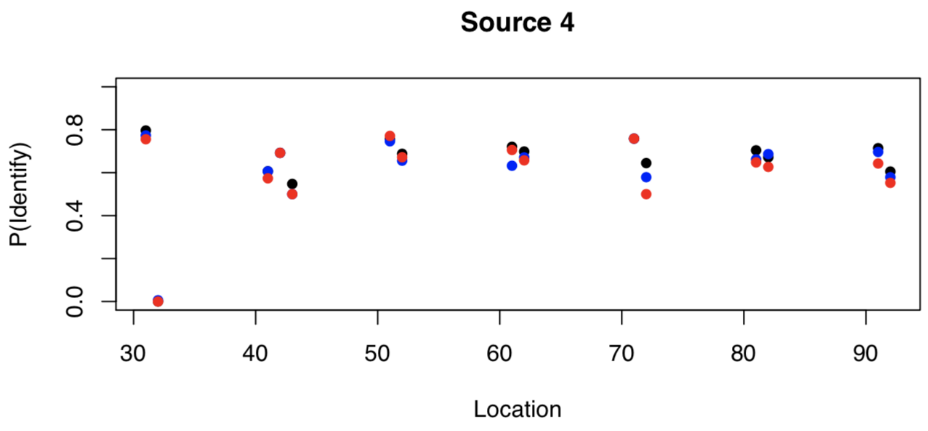
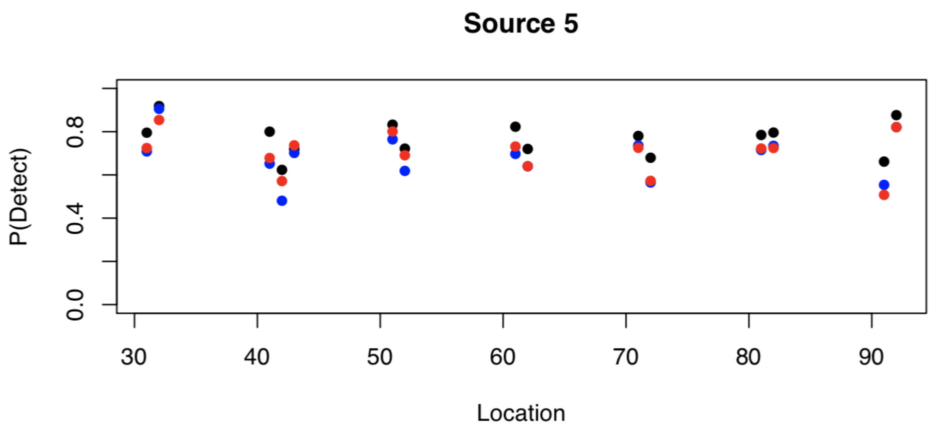


Figure 32: Proportion of correctly detected and identified runs for source 4 ($^{60}$Co). Black: 1st place, Blue: 2nd place, Red: 3rd place.



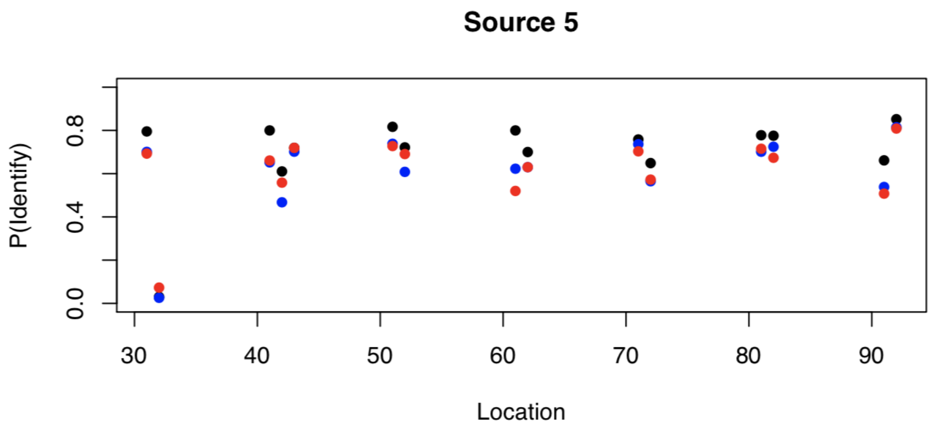
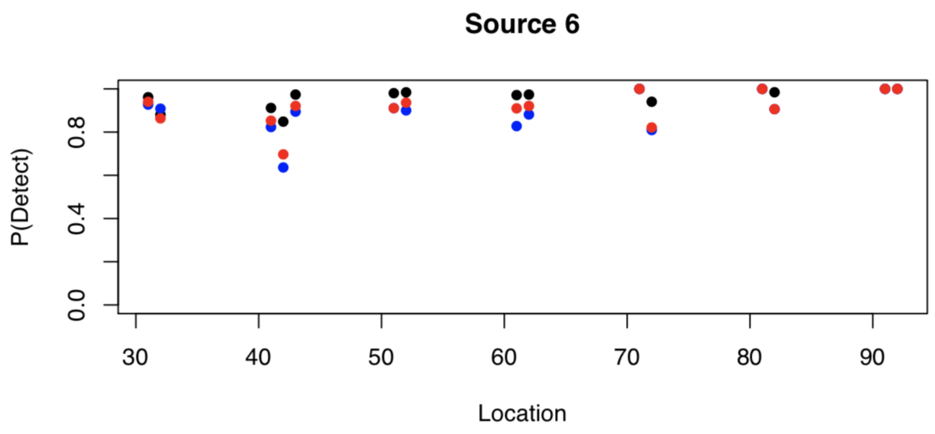


Figure 33: Proportion of correctly detected and identified runs for source 5 ($^{99m}$Tc). Black: 1st place, Blue: 2nd place, Red: 3rd place.



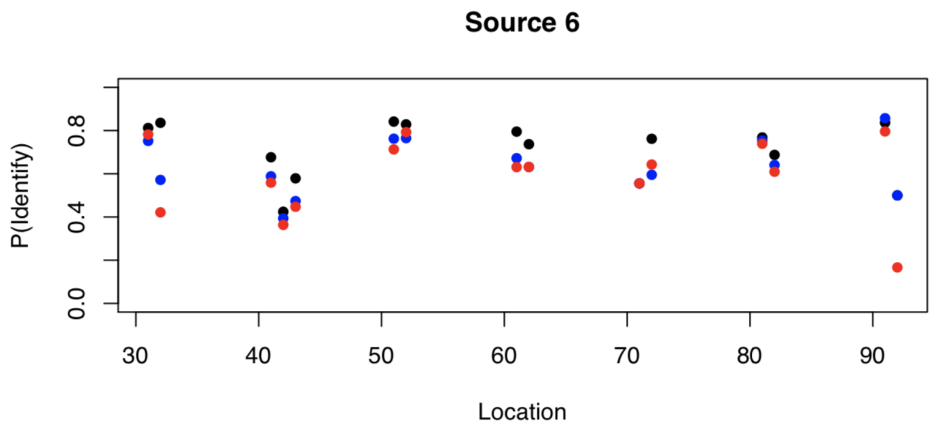


Figure 34: Proportion of correctly detected and identified runs for source 6 (HEU + $^{99m}$Tc). Black: 1st place, Blue: 2nd place, Red: 3rd place.

When we examine the results of the proportion of correctly detected and identified runs for each of the top competitors, there are several interesting results. There are clearly differences in how the algorithms perform for the different sources across the different locations. The overall summary in each plot which shows the proportion of correct detection and classification provides helpful feedback for the strengths and weaknesses of the algorithms for each source. The winning competitor is not uniformly best for all sources.

For source 1 (HEU), all 3 of the top competitors performed similarly well across all locations when it came to just the detection portion. The fraction or detected sources ranged between approximately 40% and 80%, with the first place competitor outperforming the other two at nearly every location. For identification however, the performance was relatively stable across all locations except for location 32, where the first place competitor only had a correct identification fraction of roughly 10% and the other algorithms scoring between 30% and 40%.

For source 2 (WGPu), all three competitors scored similarly across all 15 locations for both the detection and identification problems. The correctly detected fraction was relatively stable at approximately 50%, with the first place algorithm again outperforming the other two at all locations. Correct identification fraction also landed around 50%, with the first place algorithm coming out as best.

For detection of source 3 ($^{131}$I), the same trend as was seen for source 1 was found here. For identification however, all 3 algorithms performed very poorly for location 32, with correct identification fractions between 0% and 5%. This same trend is found for sources 4 ($^{60}$Co) and 5 ($^{99m}$Tc), with very poor detection and identification performance for source 4 and poor identification performance on source 5.

On the other hand, all three of the top competitors performed very well on detection of source 6 (HEU + $^{99m}$Tc), with the majority of correct detection fractions for the various locations between 90% and 100%. Likewise, most of the correct identification fractions were over 60%, with the exception of locations 32 and 92, which did not perform as well.

There are clearly easier and harder locations, as was intended in the simulation design, and this shows considerable variability around the overall proportion for the entire private data set. In particular, for sources 3, 4, and 5, all three of the top competitors performed poorly on identification at location 32, with correct identification fractions of near 0%. Figures 27 and 28 highlight a likely reason as to why this occurs, where it can be seen that approximately 25% of the runs in the testing set were at location 32, with only 0.25% of the runs in the training set at this location. From a machine learning perspective, doing this gives us the opportunity to test the algorithms at a location that is undersampled in the training set, which can possibly be used as a sort of measure of location generalizability. The poor identification performance of the algorithms at this location could possibly indicate that the spectral signature of the sources at location 32, as a result of its unique model geometry, were not well represented in the training set, making it difficult for the algorithms to achieve success with identification in the test set. Despite this, the algorithms were still able to perform the detection problem very well at this location, demonstrating a level of robustness for that particular problem.

In addition there is considerable variation between the competitors on the locations as well. When it does not appear that there are 3 dots for a particular location, this indicates that the top three competitors performed so similarly that there was overlay in the plotting symbols. However, this is not that common across all of the sources and all of the locations. The relative ordering of the three competitors for most locations remains similar for most cases, but the spacing of the points can have some variability.

### 6.1.3 Logistic Regression

The previous plots represent exploratory data analysis summaries of the results. To gain further insights into the performance of the top 3 competitors, a more formal approach that considers variations in the inputs across the runs in the data sets was used. The model-based post-competition analysis used a logistic model to gain understanding about which of the inputs were most influential in changing the probability of detection or identification.

The data were again partitioned into different subsets for each of the sources, to examine results specific to the different portions of the detection and identification problems.

A logistic regression model was fit to the data to predict both the probability of detection and the probability of identification. Note that while these predicted values across the input space are called "probabilities", it is perhaps better to think of them as a smoothed summary of the "fraction of correct detections and identifications" for a particular algorithm. They are unbiased estimates of the probability of detection or identification, but may have quite large associated variances. Hence we can think of the presented results as an empirical assessment based on the observed data.

Another important consideration for the modeling is the unequal number of runs in different regions of the input space. While this strategic choice to emphasize more difficult regions of the input space for the private and public test sets serves an important role in the ranking of the competitors based on their ability to demonstrate good characterization in more challenging scenarios, it does mean that the prediction for the contours is not as precise in some regions as in others. As we interpret some of the results later in this subsection, we highlight how sparsity of data may lead to some instability in the prediction for the logistic regression.

From Figure 22, the data were categorized as a success if they contributed to the green or dark blue, when considering the probability of detection, or just the green when considering the probability of identification.

For the probability of detection, the model form was

$$P(detect) \quad = \quad e^{x'\beta}/(1+e^{x'\beta})$$

where

$$\begin{aligned} x'\beta \quad = \quad & \beta_0 + \beta_{SNR}X_{SNR} + \beta_{Shield}X_{Shield} \\ & +\beta_{Background}X_{Background} \\ & +\beta_{Lane}X_{Lane} + \beta_{Speed}X_{Speed} \end{aligned}$$

In the above equation, $X_{SNR}$ denotes the signal to ratio value for the source, while $X_{Shield}$ is an indicator variable which is 1 if the source was shielded, and 0 otherwise. $X_{Background}$ is an indicator variable which specifies which of the 8 backgrounds was used in the run. To capture characteristics of the detector's movement, $X_{Lane}$ is a continuous variable with values 1 through 4 to denote how close the lane is to the location of the source, with 1 being the closest. $X_{Speed}$ is a continuous variable that denotes the speed of the detector in meters per second.

The probability of identification model had the same form as the probability of detection. When the models were fit for each competitor's final submission, the first stage of analysis was to evaluate which of the terms in the model were statistically significant. We found that there was remarkable consistency in the results across different sources and competitors.

For both the probability of identification and detection models, $\beta_{Background}$ and $\beta_{Lane}$ model parameters were rarely significant. This indicates that which version of the street was being modeled and which lane the detector was traveling in was not a good predictor of the performance of the various algorithms. This is reassuring in that it is highly desirable for the competitors' solutions to be robust to these nuisance factors. The remaining terms in the model, involving $\beta_{SNR}$, $\beta_{Shield}$ and $\beta_{Speed}$ were consistently highly significant. The large sample size of design allowed for considerably power for the testing of significant terms, and hence should indicate confidence in having found real patterns in the data.

After this initial model fitting, more flexible higher order models were considered for only those terms which were significant. A full second order model with two-way interactions between the terms as quadratic terms for speed and signal to noise ratio were considered to allow for better fitting of the responses. The full model was fit, and then model selection performed with only those terms that were statistically significant at the 5% level retained in the model.

Figures 35 through 37 show the fitted contour plots for each of the top three competitors for source 1 (HEU). The top row of the plots shows the fitted model for the probability of detection, while the bottom row shows the probability of identification. The left side of the plot provides results for the cases when the source was not shielded, while the right side of the plot indicates results when the source was shielded. Within each plot, the x-axis shows the speed of the detector, while the y-axis indicates changes in the signal to noise ratio.

Since we would expect that the algorithms would find a fast moving detector combined with a small amount of source material to be the most challenging version of the problem, the probabilities of detection and identification are generally the smallest in the bottom right corner of each plot.

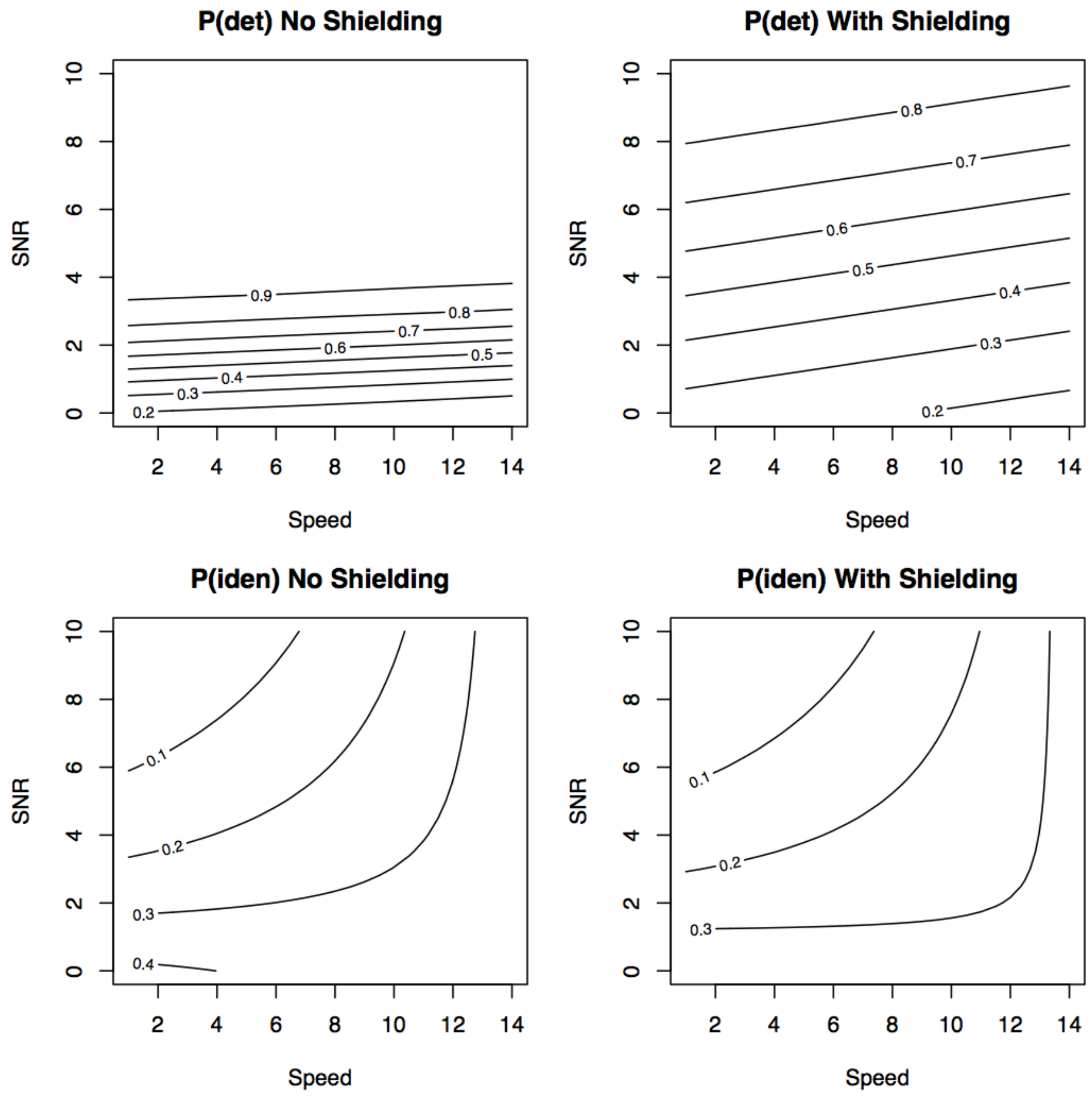


Figure 35: Contour plots for source 1 (HEU) for the winning competitor, `pfr` .

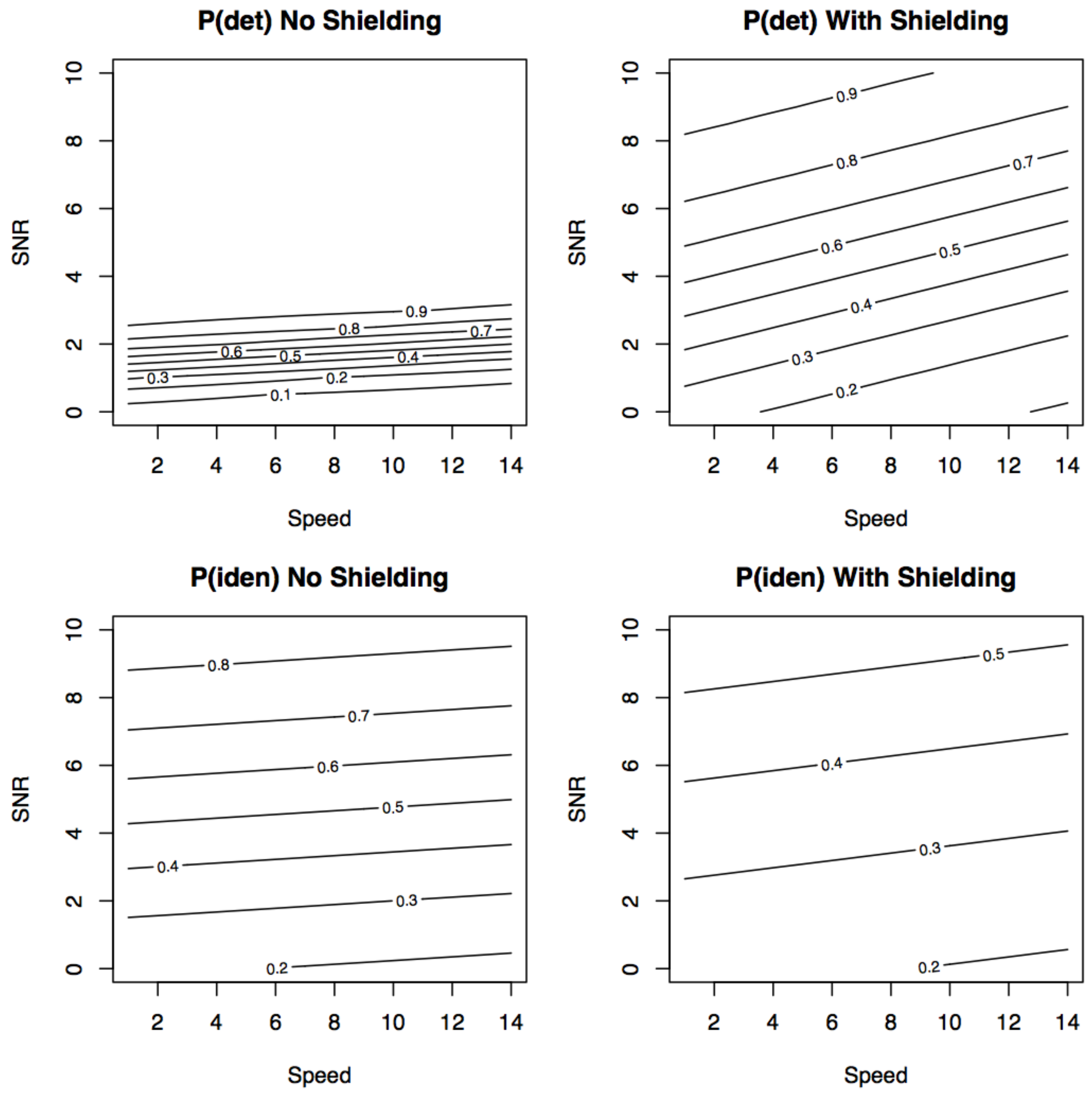


Figure 36: Contour plots for source 1 (HEU) for the 2nd place competitor, `p_kuzmin` .

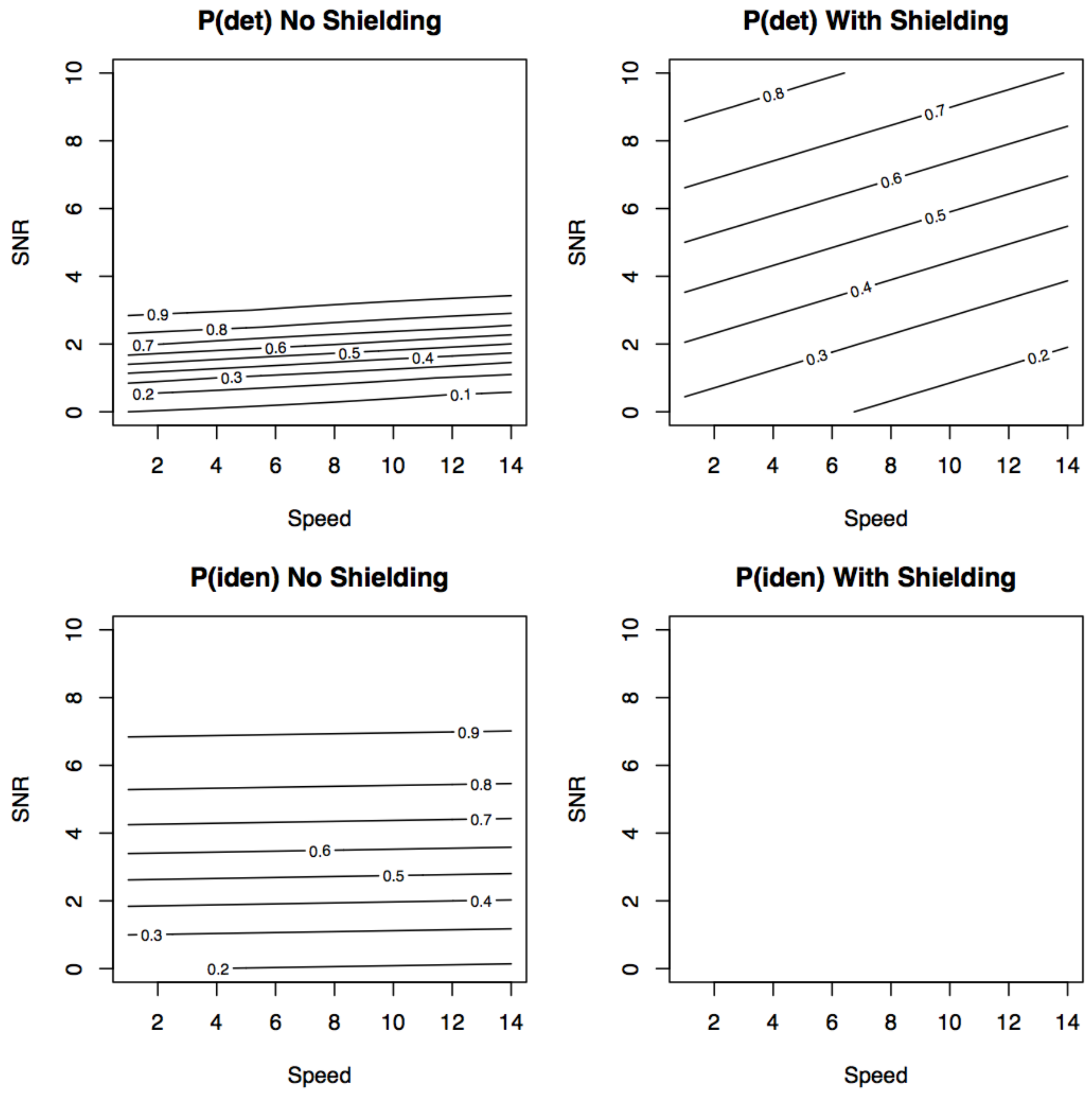


Figure 37: Contour plots for source 1 (HEU) for the 3rd place competitor, `gardn999` .

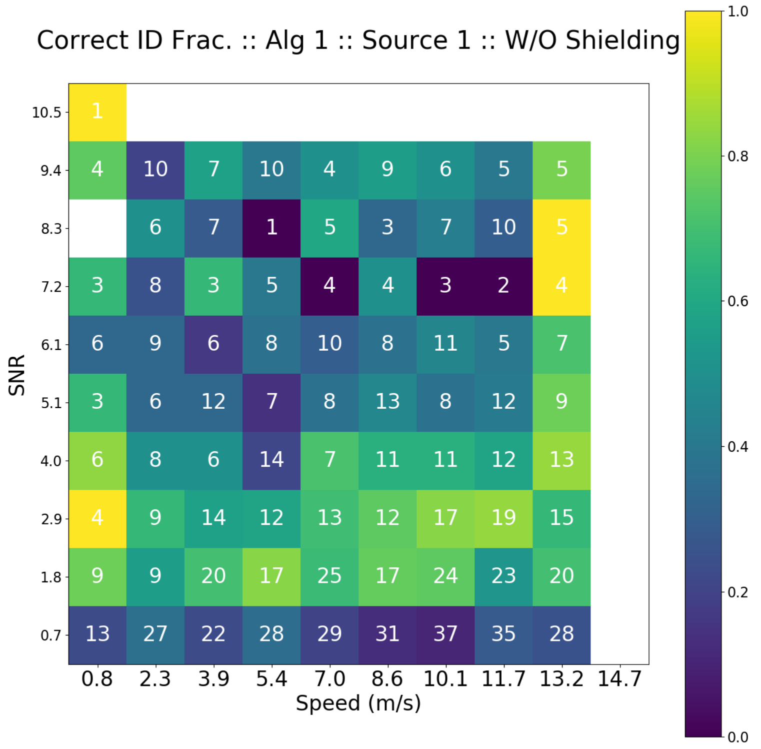


Figure 38: Heatmap for winning algorithm for unshielded source 1 (HEU) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

When we examine the results for the top three competitors for source 1 (HEU) we see that in general the unshielded version of the problem is less challenging than the shielded version. For all three competitors, SNR values greater than 4 for the unshielded source had a very high probability ($> 0.9$) of detection. For the shielded source, the results for the probability of detection indicate that the speed of the detector plays a role in the ability to get the correct answer (as denoted by the non-horizontal contour lines).

The shape of the contour curves for the probability of identification look much more varied between the competitors. The winner in Figure 35 has inverted contours that indicate that it has a more difficult time with higher signal to noise ratio values than small. This unintuitive result may be an artifact of the smaller sample sizes in the high SNR values for the test set data. Figure 38 shows the detailed breakdown of the proportion correct for different combinations of SNR and detector speed. While there were less runs in the top portion of the plot, as denoted by the smaller numbers in each cell, there does appear to be a pattern of poorer performance for some of the larger SNR values. The small sample sizes for these regions makes the variability of these estimates large, but it is clear how the logistic regression identified this pattern. If we restrict ourselves to a smaller portion of the region, where there is more data (say in the bottom right portion with $> 10$ runs per combination, then we see that there is a more expected pattern of poorer performance for very low SNR and some improvement for larger values (say up to SNR of 2.9).

When we reexamine Figure 22, we see that the winner performed well on detection for source 1 (HEU), but was not one of the top performers for the identification aspect. Both the second and third place finishers has good identification ability for the no shielding case with a high probability of correct identification as the SNR value gets larger. The ability of these two competitors to get the identification correct is much more

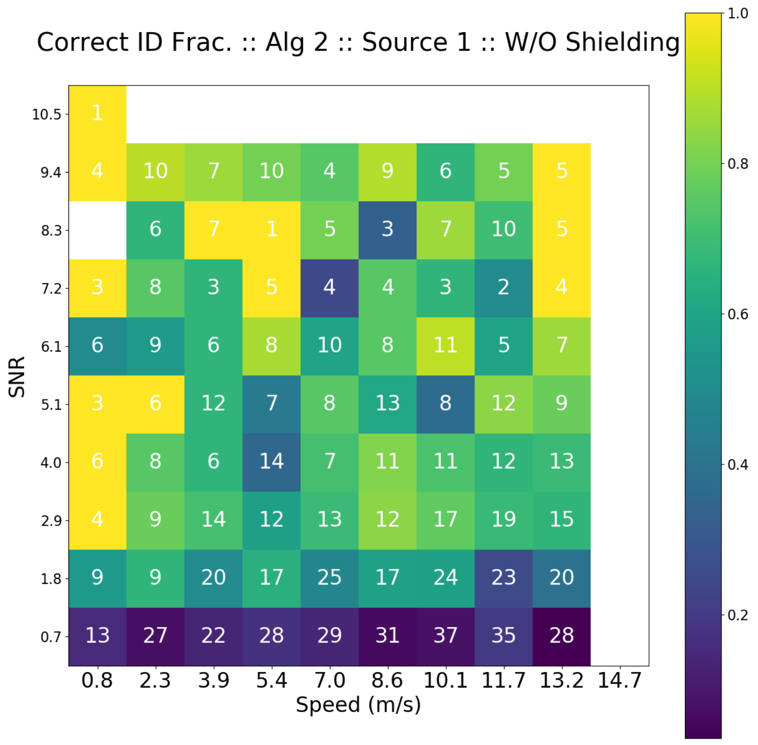


Figure 39: Heatmap for 2nd place algorithm for unshielded source 1 (HEU) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

dependent on the SNR than the speed of the detector, as can be seen by the near horizontal nature of the contour lines. Figure 39 shows a corresponding heat map for the second place finisher. When we compare it to Figure 38, we see that this has a much more typical pattern with improving performance as we move to the top left corner of the plot.

Overall, the second and third place competitors perform better than the overall winner for identification of source 1 (HEU). All three competitors perform similarly well for the detection aspect of the problem, with the shielded source being more challenging.

From Figure 22, we see that source 2 (WGPu) was one of the easier sources to detect and identify. When we examine the contour plots for the top three competitors in Figures 40 through 42, we see some common patterns emerging.

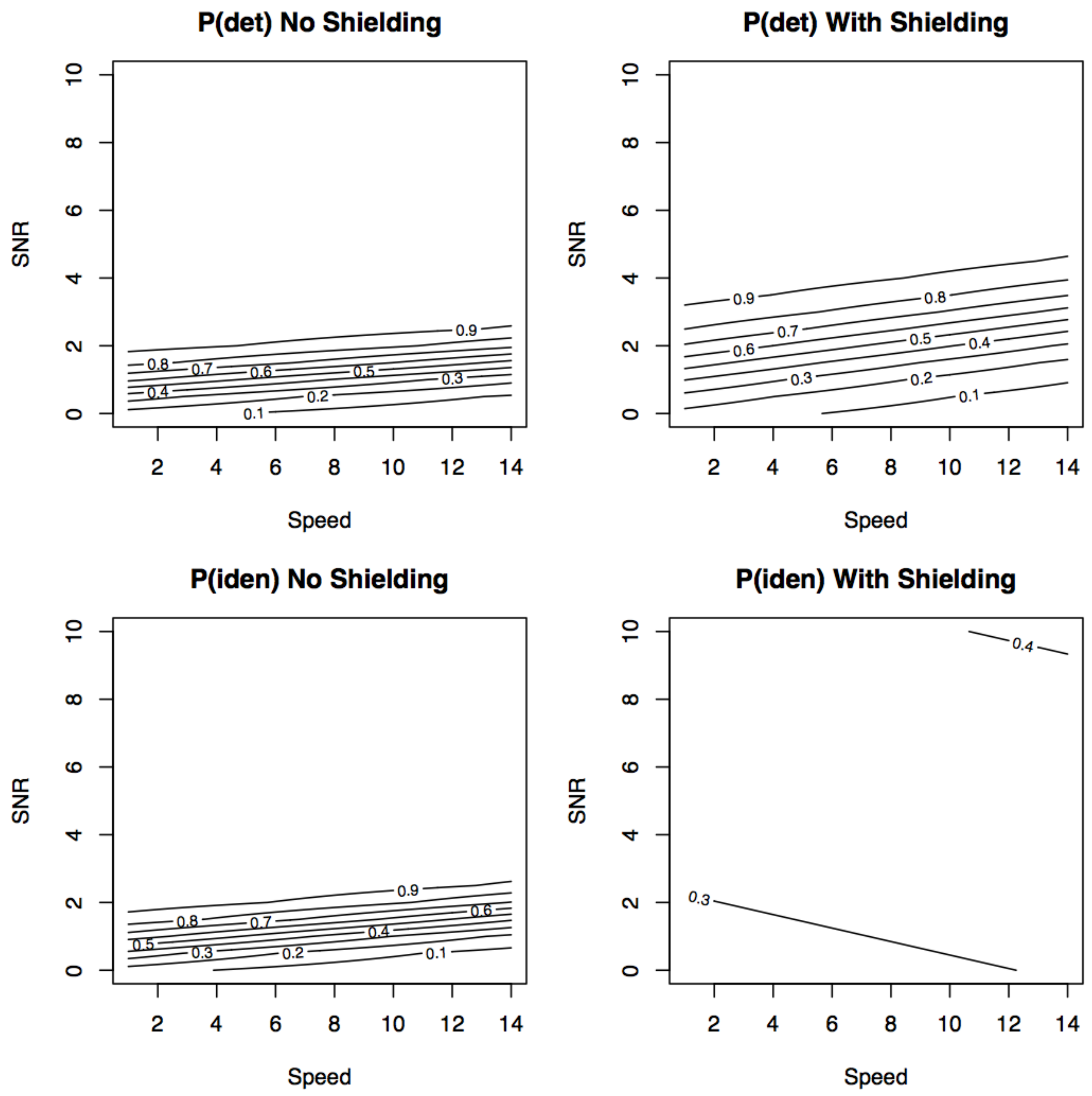


Figure 40: Contour plots for source 2 (WGPu) for the winning competitor, `pfr` .

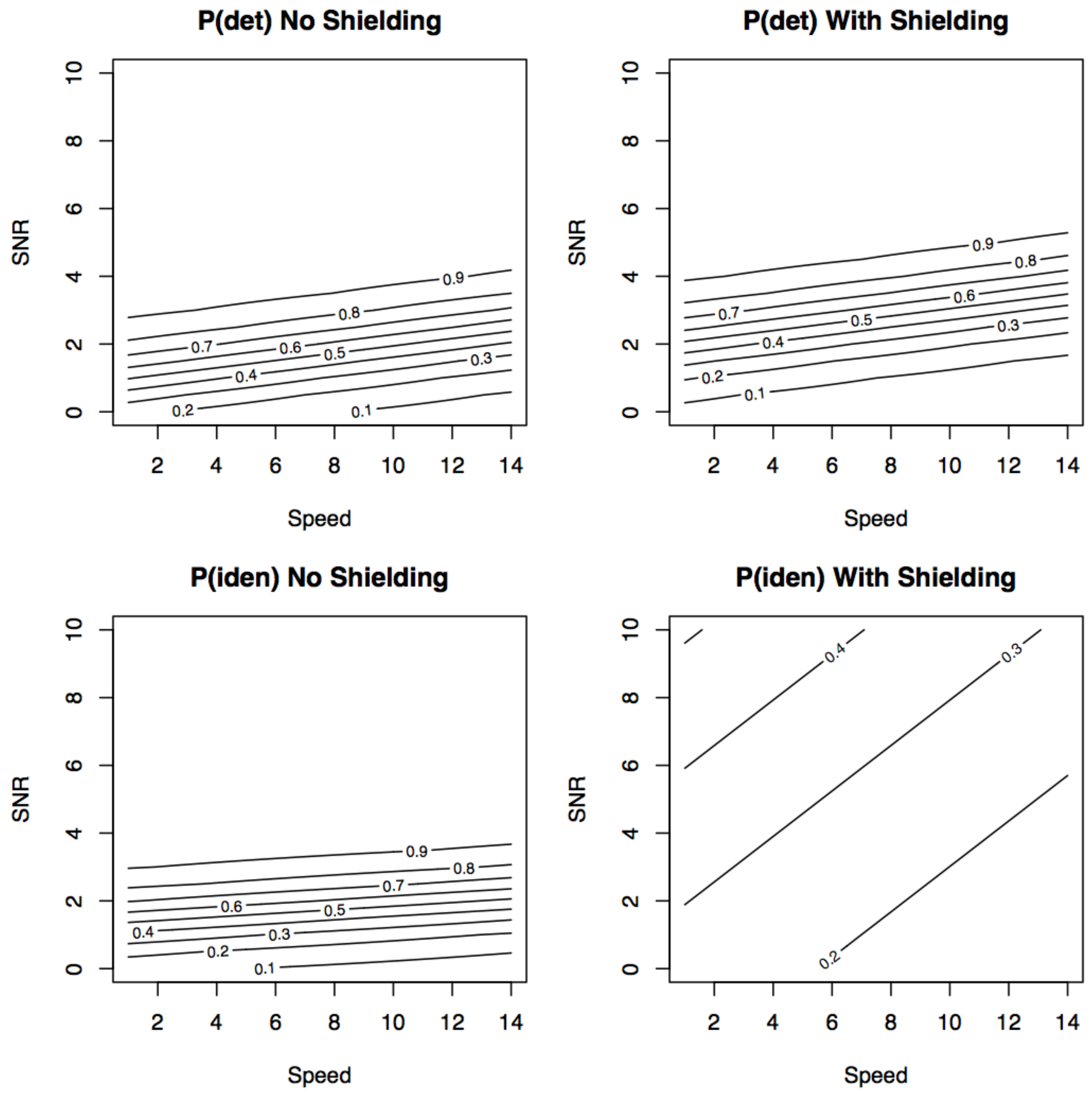


Figure 41: Contour plots for source 2 (WGPu) for the 2nd place competitor, `p_kuzmin` .

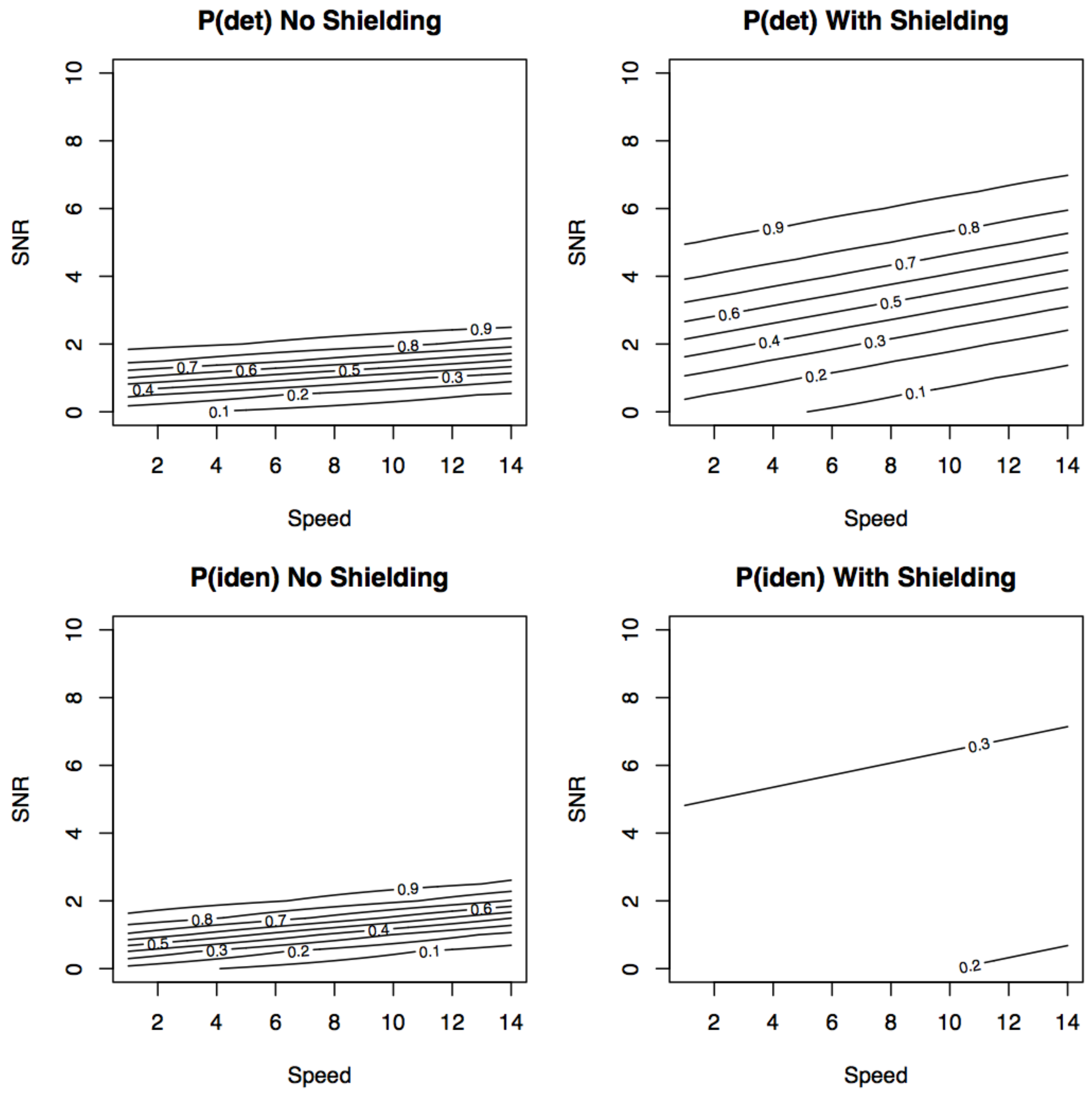


Figure 42: Contour plots for source 2 (WGPu) for the 3rd place competitor, `gardn999` .

The resulting contour plots from the logistic regression models for source 2 (WGPu) in Figures 40 through 42 show that for both the probability of detection and identification when the source is unshielded, all three of the top competitors are able to have a high probability of success when the SNR ratio is larger than 3. The speed of the detector does not seem to be a major contributor to differences in performance for this source when there is no shielding as the contour lines are close to horizontal for all top competitors.

For detection, the shielded source poses slightly more challenges, but overall for SNR values greater than 4, all of the top three competitors were able to do well.

There are differences in the probability of identification for the shielded source, with none of the competitors being able to achieve a high probability of success. The surface is relatively flat with a narrow range of overall correct identification throughout the region.

The contour plots for source 3 ($^{131}$I) show some interesting patterns across the top three competitors as illustrated in Figures 43 through 45.

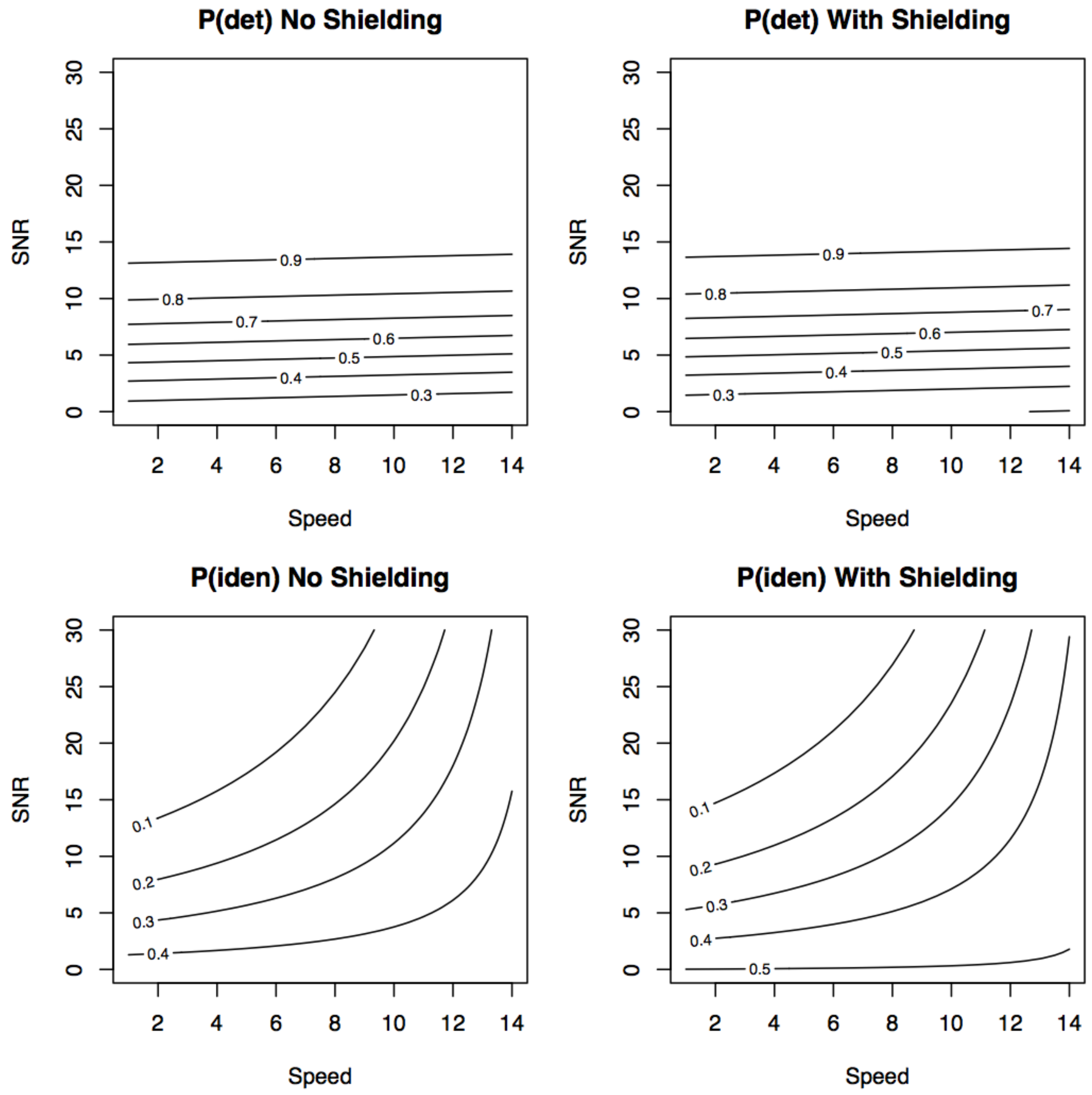


Figure 43: Contour plots for source 3 ($^{131}$I) for the winning competitor, `pfr` .

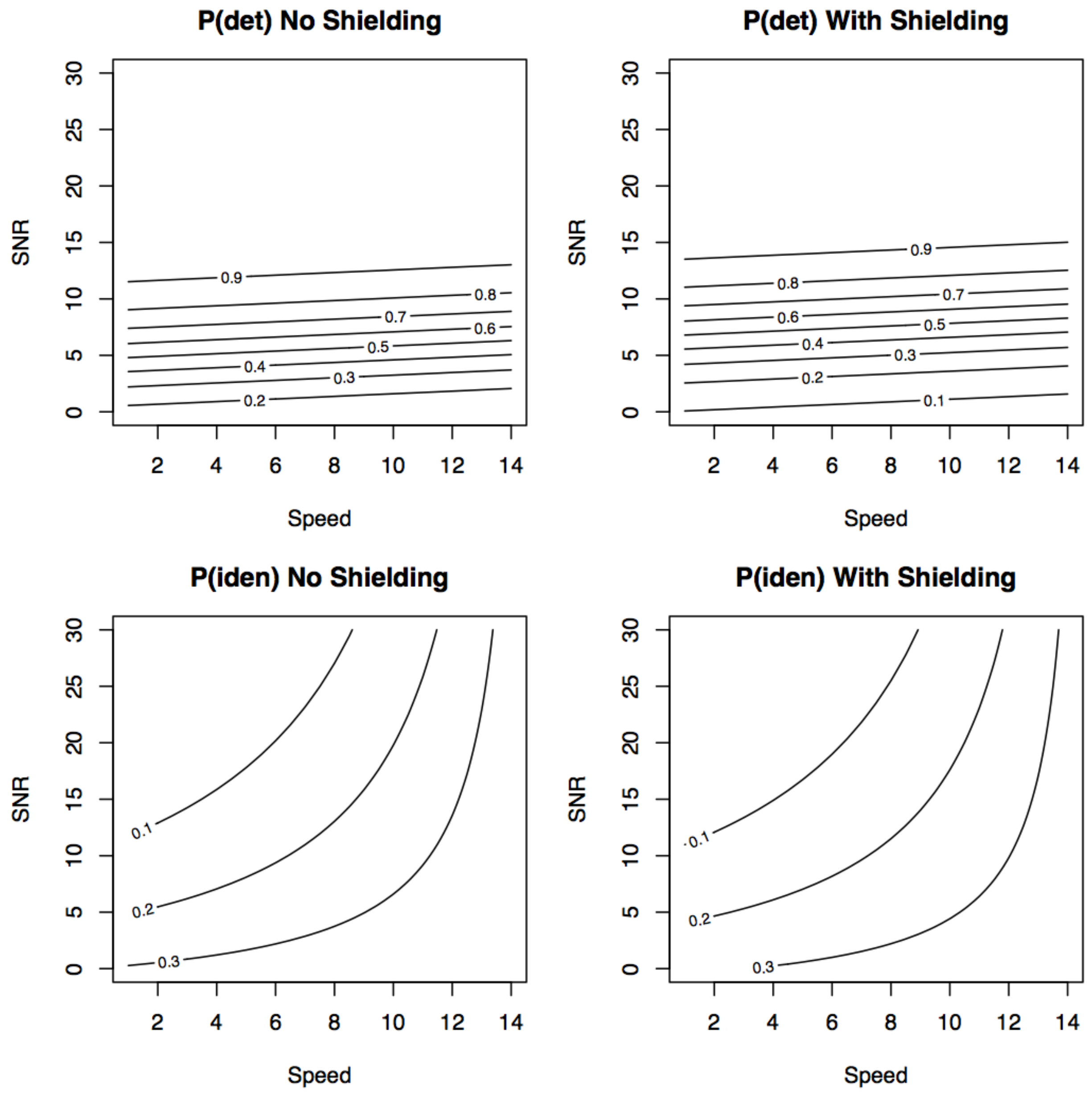


Figure 44: Contour plots for source 3 ($^{131}$I) for the 2nd place competitor, `p_kuzmin` .

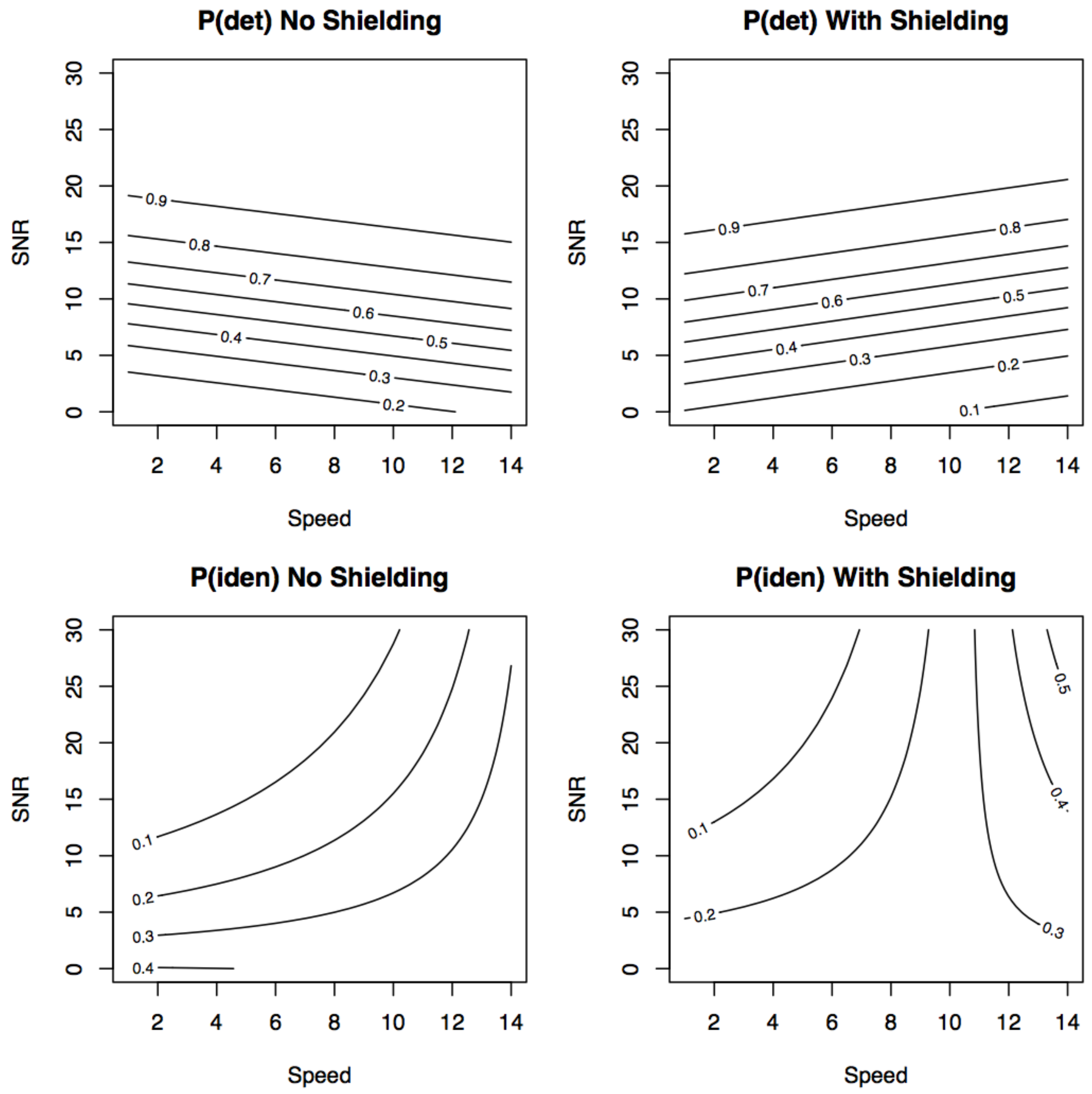


Figure 45: Contour plots for source 3 ($^{131}$I) for the 3rd place competitor, `gardn999` .

The resulting contour plots from the logistic regression models for source 3 ($^{131}$I) in Figures 43 through 45 show some similarities between the first and second place competitors, while the pattern for detection and identification for third place finisher looks somewhat more different.

For the first and second place finishers, there is relatively little difference in performance for the probability of detection whether the sources was shielded or not. Again, the SNR has a much greater impact on detection than the speed of the detector (nearly horizontal lines).

For the third place finisher, there are more differences between the shielded and unshielded versions of the problem, with the pattern in the data indicating that this algorithm is slightly better at detection for the unshielded case and a fixed SNR value when the detector is moving more quickly. The more typical pattern of detection being harder with faster detector speeds is observed for the shielded subset of data.

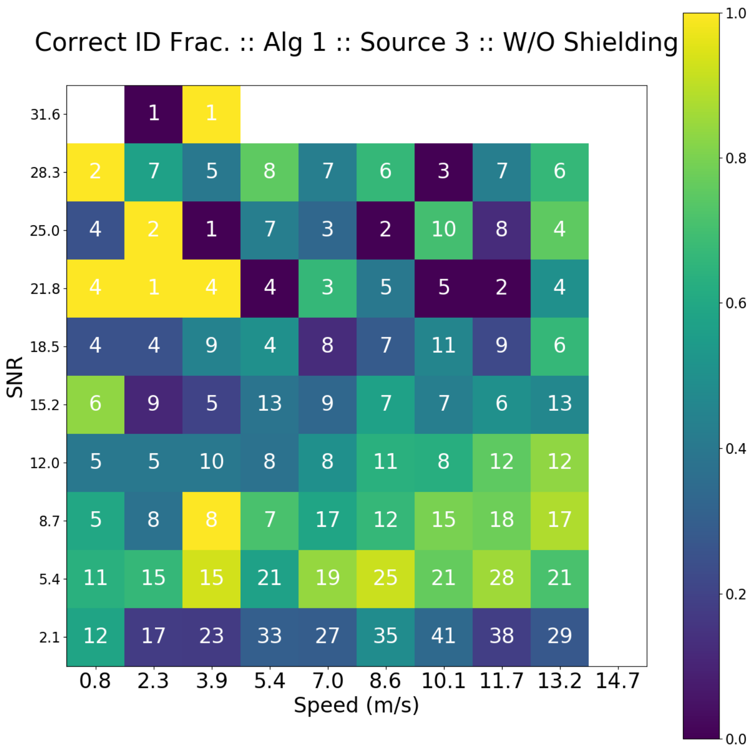


Figure 46: Heatmap for winning algorithm for unshielded source 3 ($^{131}$I) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

For identification, all three top competitors have a more difficult time with this aspect of the problem than detection. The contours indicate a relatively low success rate throughout the region with larger SNR values proving more challenging. Again the small sample sizes for the number of runs in the test set with large SNR values, leads to some larger variances for the local estimates of the fraction correct, which translates into some instability in the surface fitting.

Figures 46 and 47 provide a more detailed look at the pattern of correct identification for both the unshielded (Figure 46) and shielded (Figure 47) versions of source 3 ($^{131}$I).

When we examine the figures, we again see patches with very low probability of correct identification for larger SNR values, but they occur in cells with very small sample sizes. For the lower portion of the plots (where we have double-digit numbers of runs per cell), we have low success rates for the very smallest SNR values, and then some improvements as the SNR value improves. When we move to the top half of the plot, we start to see much less consistency in the results for adjacent cells. Similar patterns are observed for the 2nd and 3rd place algorithms as well.

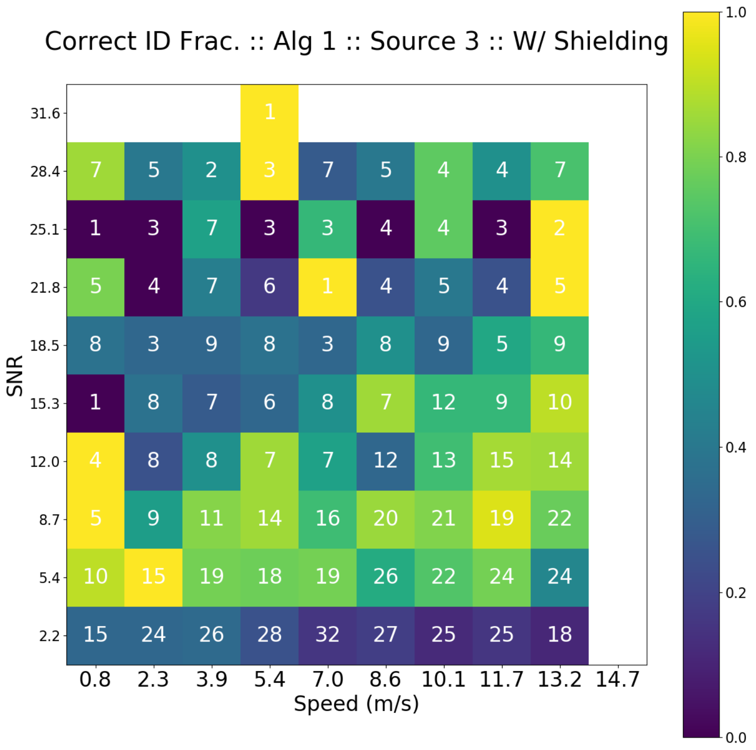


Figure 47: Heatmap for winning algorithm for the shielded source 3 ($^{131}$I) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

The contour plots for Source 4, cobalt-60, show some interesting patterns across the top three competitors as illustrated in Figures 48 through 50.

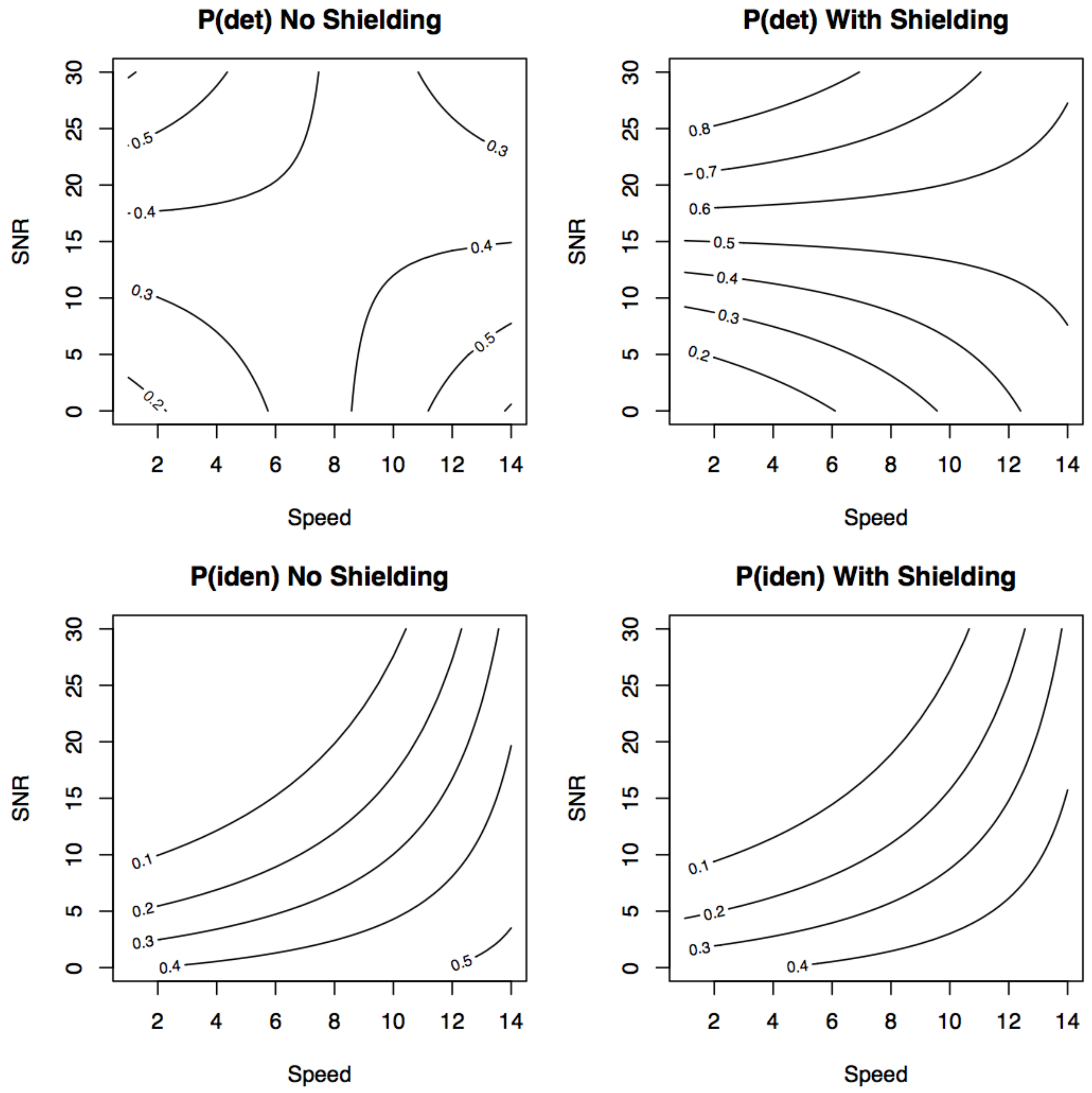


Figure 48: Contour plots for source 4 ($^{60}$Co) for the winning competitor, `pfr` .

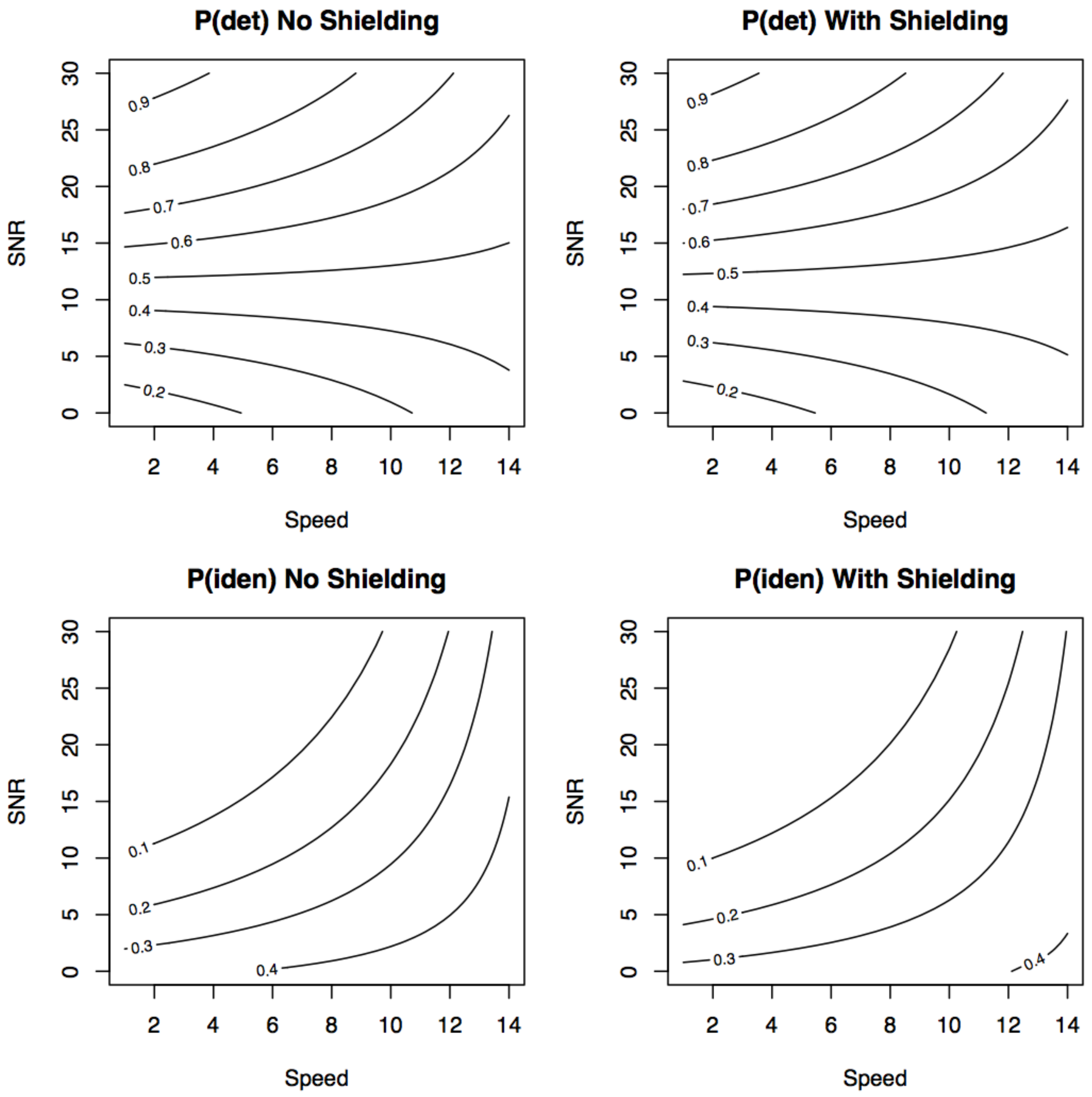


Figure 49: Contour plots for source 4 ($^{60}$Co) for the 2nd place competitor, `p_kuzmin` .

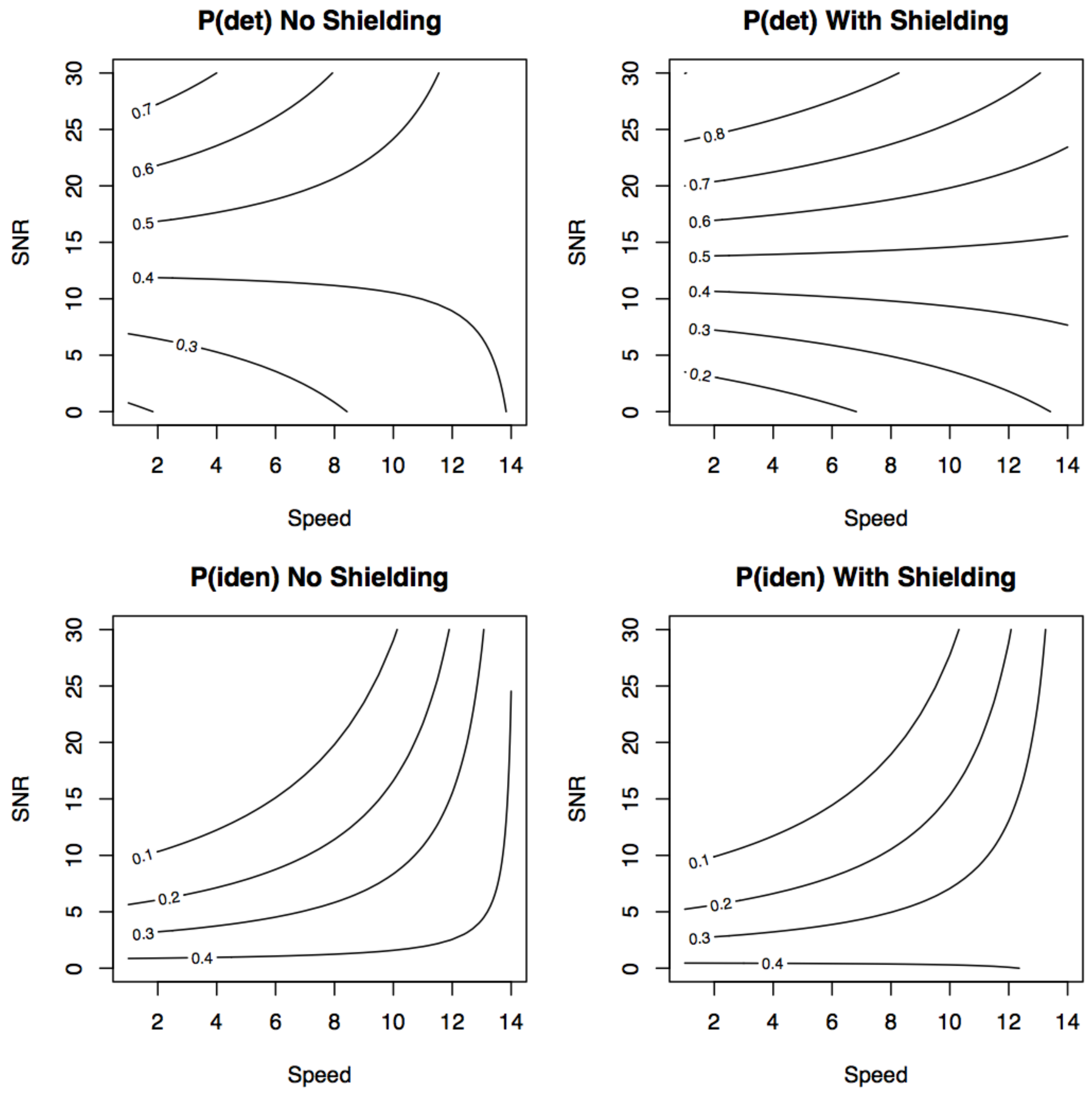


Figure 50: Contour plots for source 4 ($^{60}$Co) for the 3rd place competitor, `gardn999` .

When we examine the results of the logistic regression fitted models for source 4 ($^{60}$Co) in Figures 48 through 50, we see some different patterns than for some of the other sources.

All three of the competitors exhibit different patterns in their ability to detect this source across the combinations of SNR and speed of the detector. In other sources, we have seen near horizontal lines for the contours for both the shielded and unshielded sources. Here, we see some more complicated patterns with different features as the speed of the detector increases to near maximum speed. For all three competitors, the best performance for detection occurs in the top left corner of the plot with minimum speed and maximum SNR. The region where the probability of detection exceeds 0.9 is very small and in some cases is not achieved by the algorithm.

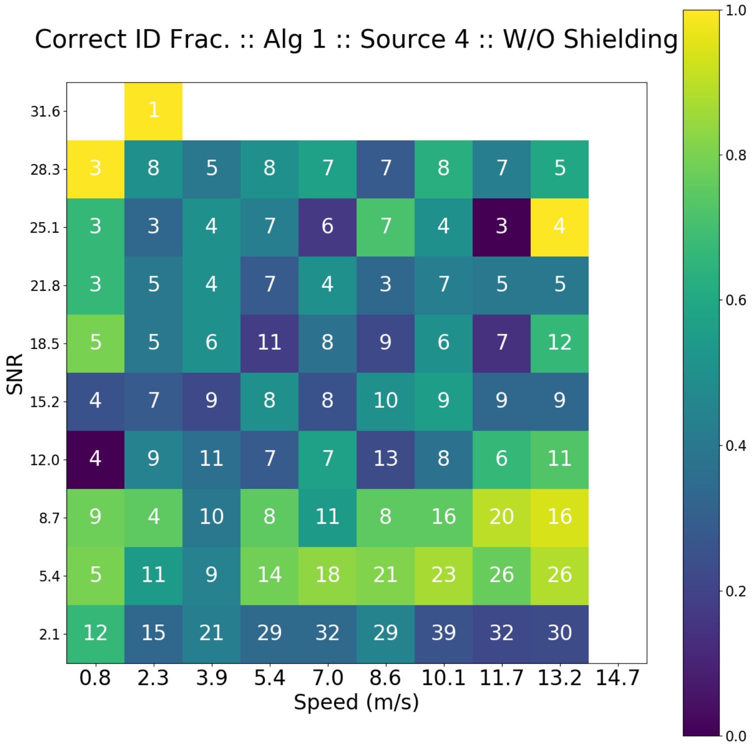


Figure 51: Heatmap for winning algorithm for unshielded source 4 ($^{60}$Co) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

The predicted probability of correct identification again seems less than intuitive, with all three competitors having the most success for both the shielded and unshielded cases when the SNR values are small. As with Sources 1 and 3, the explanation for this behavior may be partially a result of the nature of the spread of the runs for the test set across the ranges of the SNR and speed inputs.

Figures 51 and 52 provide a more detailed look at the pattern of correct identification for both the unshielded (Figure 51) and shielded (Figure 52) versions of tsource 4 ($^{60}$Co) for the winning competitor.

The figures show quite a few cells with quite observed low success rates, and we see that there is generally a lack of systematic patterns connecting the results from adjacent cells in the plots. As a result the logistic regression and estimated models struggle to characterize the patterns present in the data. The most consistent results appear in the bottom right corner of the plots where we have larger numbers of runs for

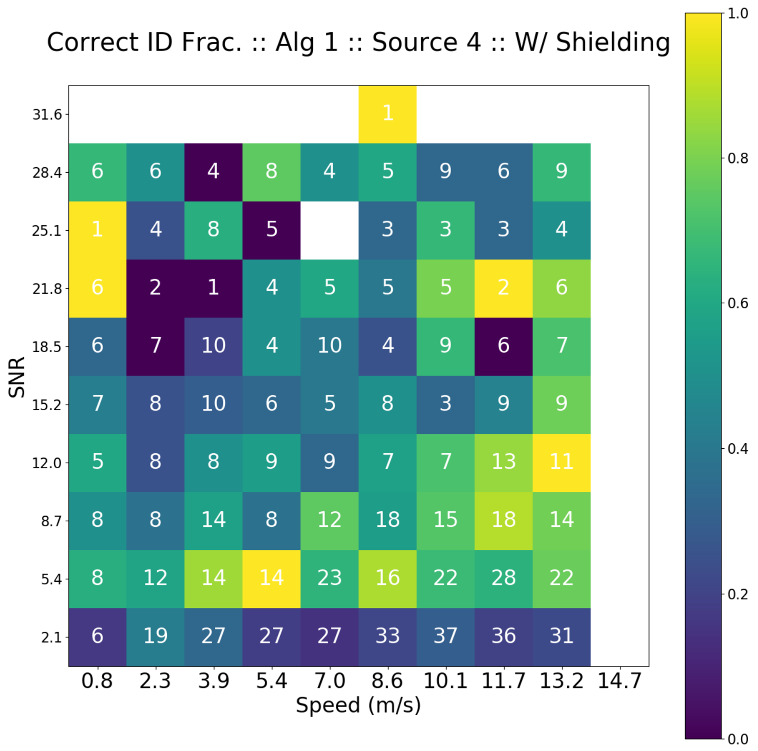


Figure 52: Heatmap for winning algorithm for the shielded source 4 ($^{60}$Co) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

each of the cells. This is to be expected since the variability of the fraction correct will stabilize more with increasing sample size. Similar patterns are observed for the 2nd and 3rd place algorithms as well.

Figures 53 through 55 show the results for the probability of detection and identification for source 5 ($^{99m}$Tc).

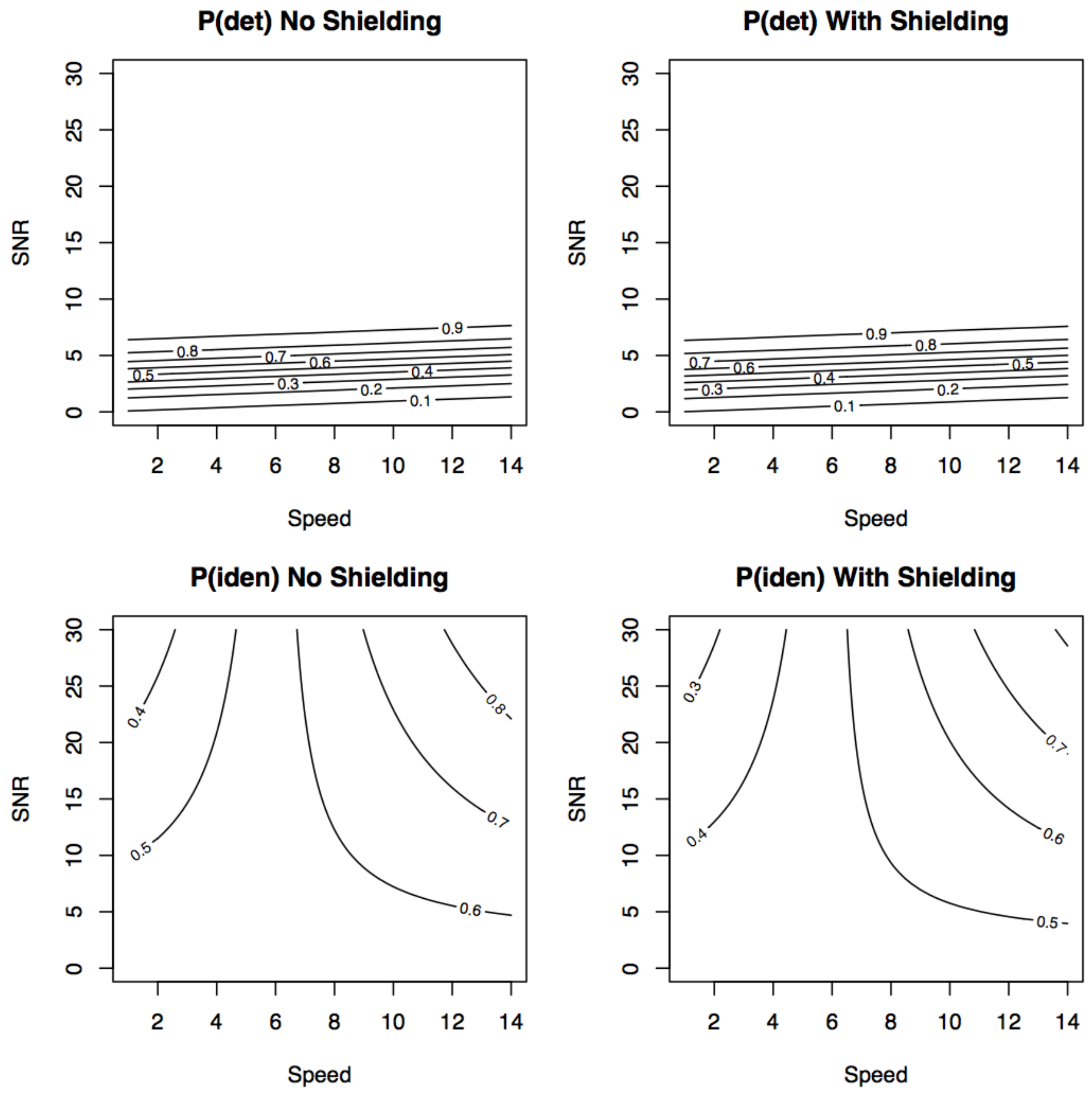


Figure 53: Contour plots for source 5 ($^{99m}$Tc) for the winning competitor, `pfr` .

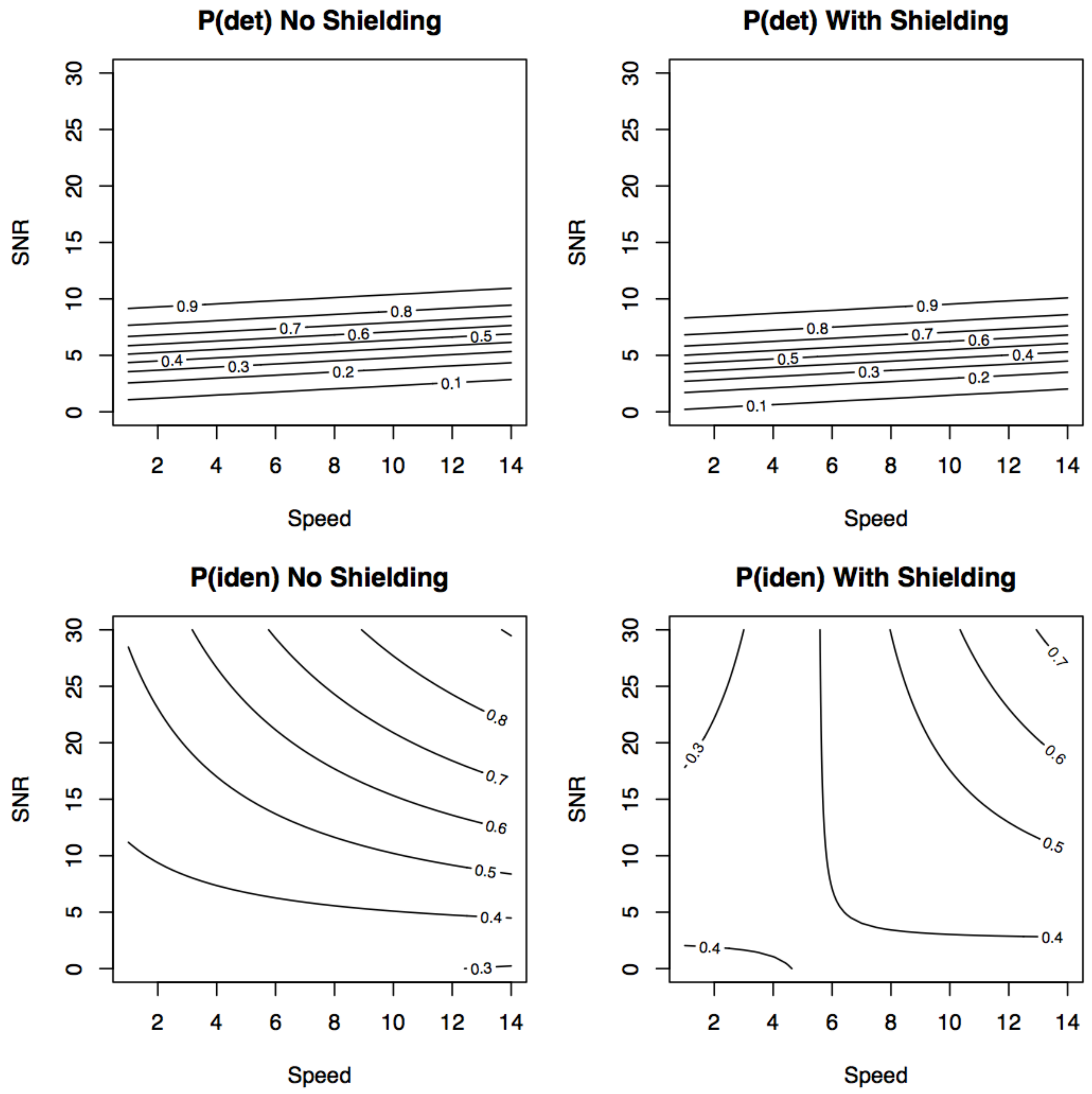


Figure 54: Contour plots for source 5 ($^{99m}$Tc) for the 2nd place competitor, `p_kuzmin` .

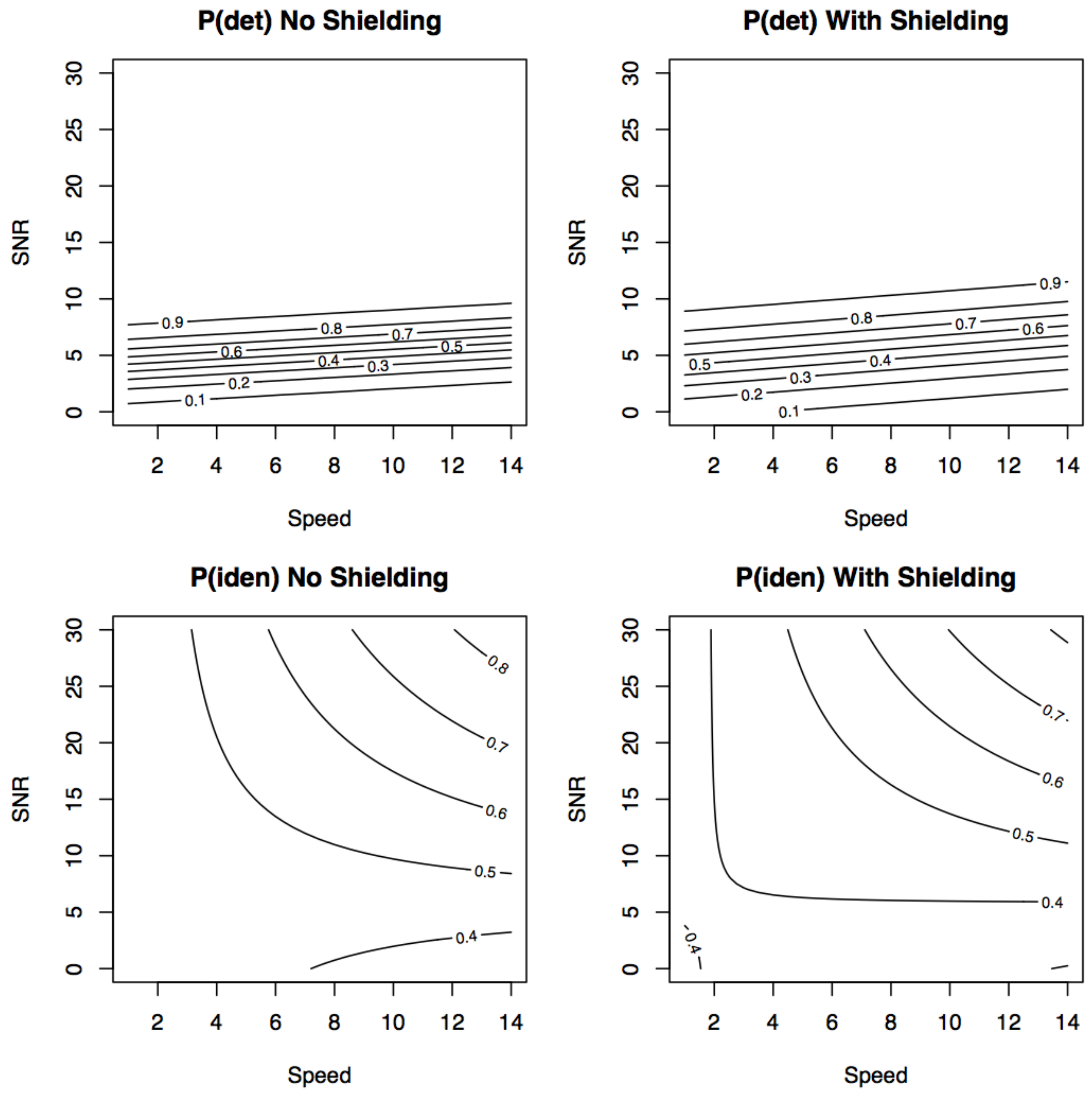


Figure 55: Contour plots for source 5 ($^{99m}$Tc) for the 3rd place competitor, `gardn999` .

When we examine the results for source 5 ($^{99m}$Tc) in Figures 53 through 55, we see some more familiar patterns.

All three competitors have a high probability of correct detection when the SNR value is large (say, greater than 8). The near horizontal lines of the contour plots for both the shielded and unshielded versions indicate that the speed of the detector has minimal impact on their probability of correctly detecting this source.

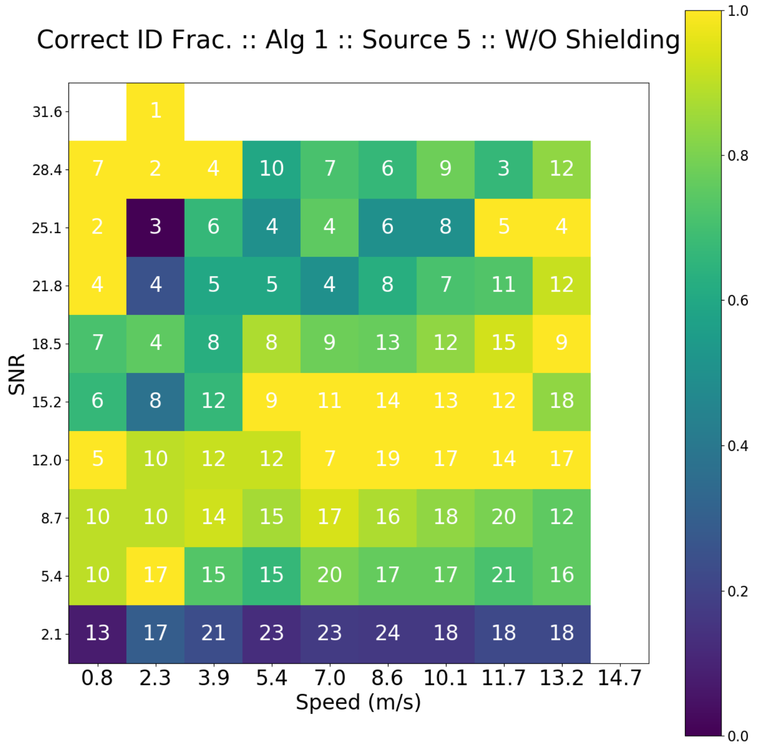


Figure 56: Heatmap for winning algorithm for unshielded source 5 ($^{99m}$Tc) proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

The patterns for the probability of correct identification for the three competitors exhibit considerably differences with some of the higher speeds appearing to have better success rates. Again, it is possible to gain some insights into how these patterns emerge by examining the heatmaps for the probability of correct identification more locally within the region. Figures 56 and 57 provide a more detailed look at the pattern of correct identification for both the unshielded (Figure 56) and shielded (Figure 57) source 5 ($^{99m}$Tc) for the winning competitor.

The overall fraction correct in the heatmaps are better than was observed for some of the other sources, but still there is a lack of systematic pattern connecting adjacent cells. Hence, this leads to different patterns in the contour plots based on the estimated logistic regression models. The lack of consistent patterns across the input space defined by SNR and speed of the detector is also reflected in the heatmaps for the 2nd and 3rd place competitors.

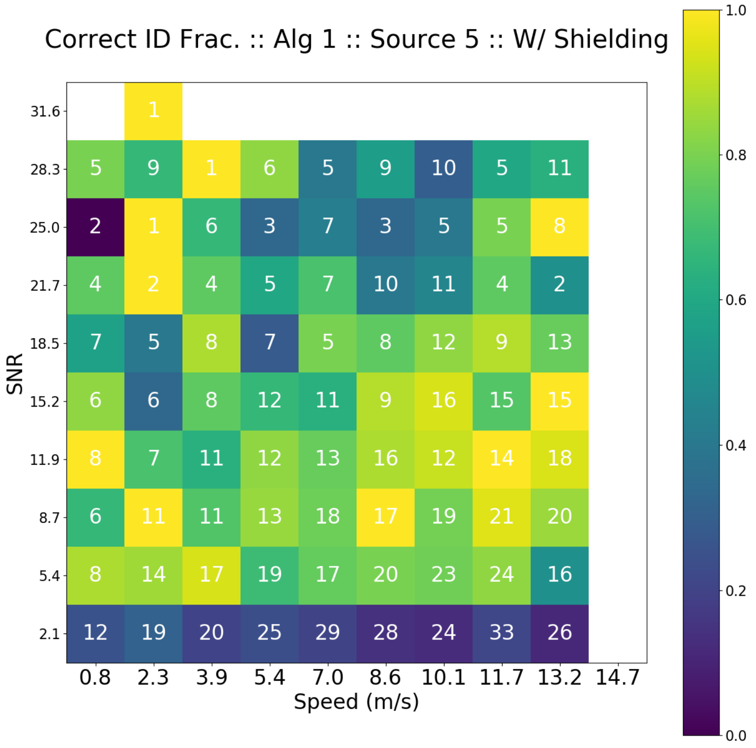


Figure 57: Heatmap for winning algorithm for the shielded source 5 ($^{99m}$Tc) where colors indicate the local proportion correct and the numbers inside the cells indicate the number of observations for that combination of speed and SNR

The final source is the mixture of HEU and $^{99m}$Tc. The summaries of the probability of detection and identification are shown in Figures 58 through 60.

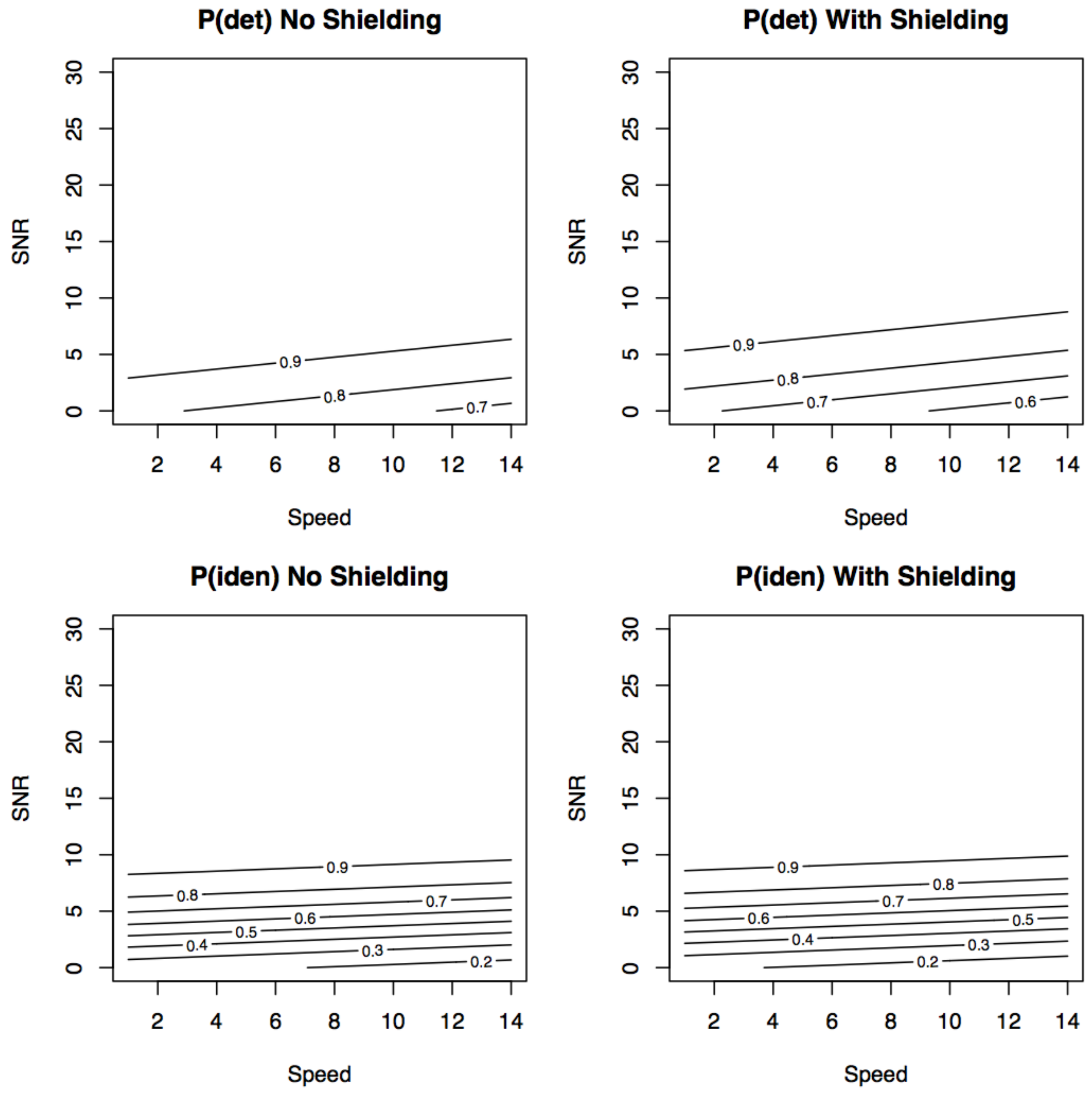


Figure 58: Contour plots for source 6 (HEU + $^{99m}$Tc) for the winning competitor, `pfr` .

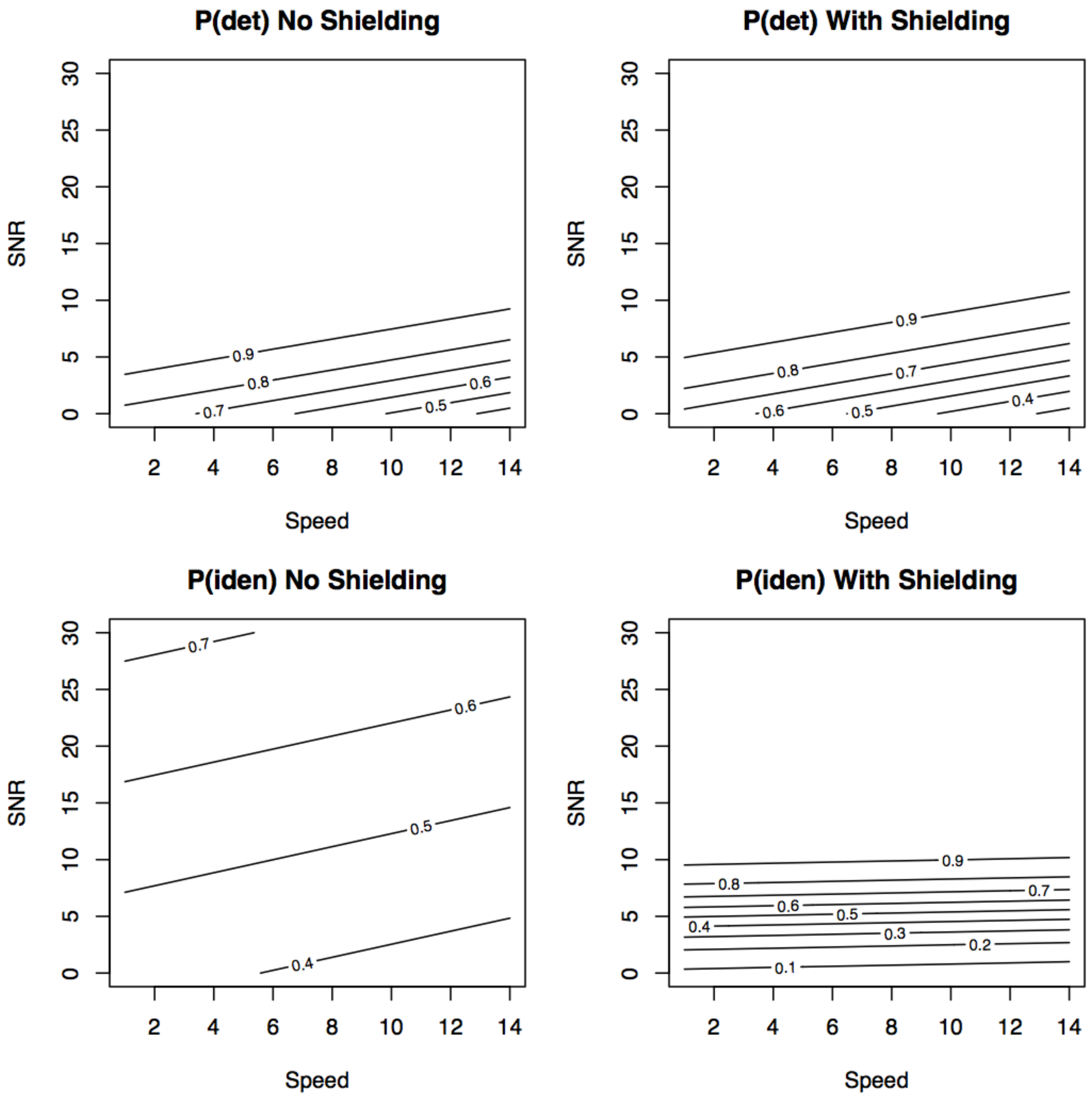


Figure 59: Contour plots for source 6 (HEU + $^{99m}$Tc) for the 2nd place competitor, `p_kuzmin` .

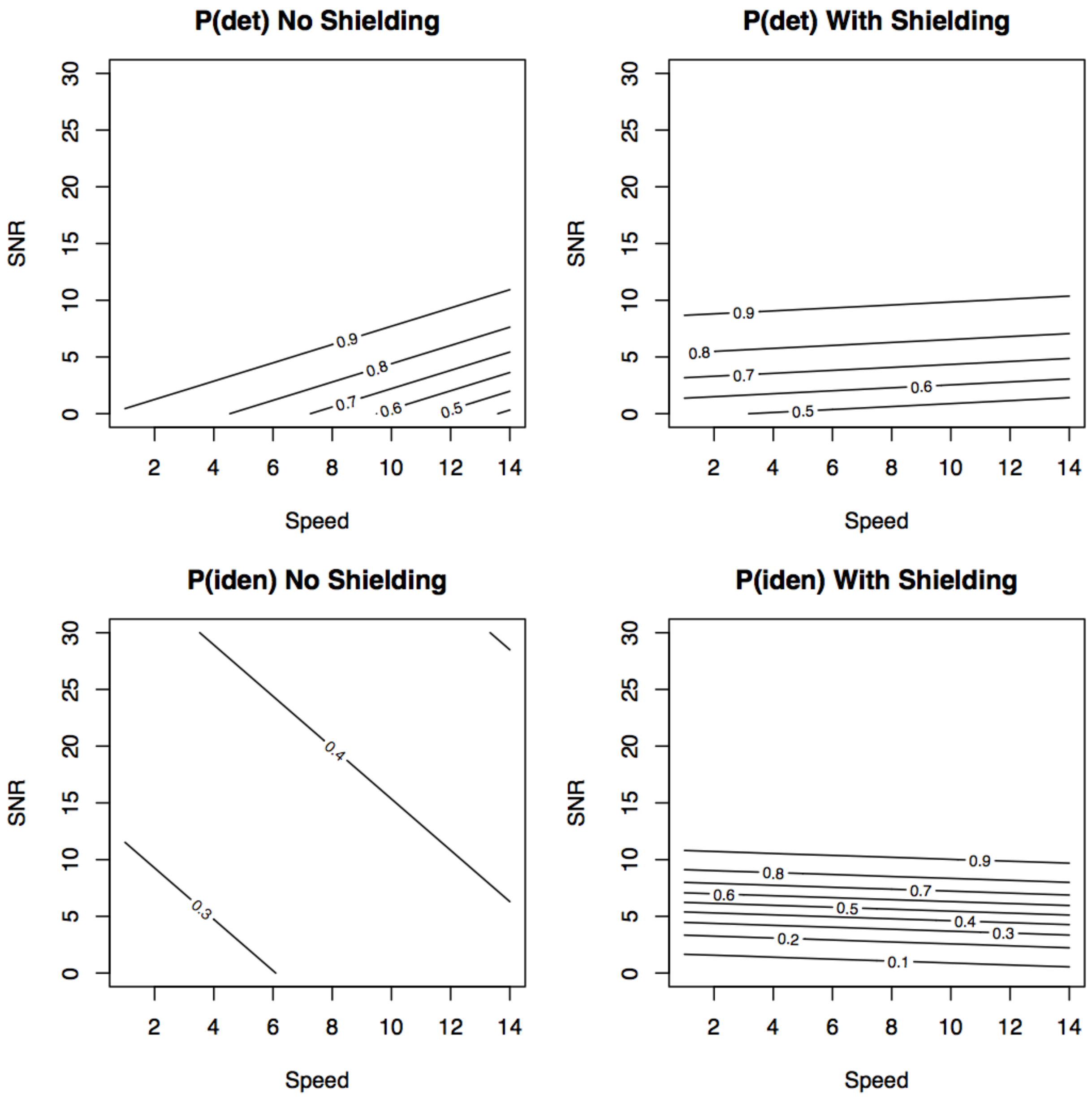


Figure 60: Contour plots for source 6 (HEU + $^{99m}$Tc) for the 3rd place competitor, `gardn999` .

Figures 48 through 50 show the results of the top three competitors for the combined source 6 (HEU + $^{99m}$Tc). The probability of correct detection is the highest for this source for the top competitors, with values exceeding 0.9 for the majority of the region. The contour lines are less horizontal for the unshielded source, with high probabilities for even very small SNR values if the speed of the detector is sufficiently slow.

For the shielded source, the contour lines are closer to horizontal indicating that the speed of the detector makes less difference for this case. With respect to identification, the winning competitor has excellent performance for SNR values greater than 10. There seems to be little difference in these surfaces regardless of whether the source was shielded or not. For the second and third place competitors, the shielded version of the problem appears to have been more easily solved, with both of them enjoying good success for SNR values greater than 10 or 12. However, both of them seemed to struggled more with the unshielded problem, and were not able to achieve high correct probabilities anywhere in the region defined by the SNR and speed of the detector.

These summaries when combined with understanding of the algorithm approaches can potentially provide insights into how the different methods used in the solutions translate into strengths for some aspects of the overall radiation detection problem.

## 6.2 Non-Uniform Space Filling Designs

A new type of space filling design, called a Non-Uniform Space Filling (NUSF) design, was developed [88]. The approach allows the concentration of points in the input region defined by the model parameters to be varied to reflect different emphases of regions. The method uses a weighted distance function that adjusts the effective weights of each of the candidate points. For the data competition, a baseline algorithm was run on a large superset of data that would later serve as the candidate set from which designs were constructed. A model of the probability of correct detection and identification for each of the sources across the range of inputs was used to estimate the weights for each candidate point. Figure 61 shows an example of the contour lines for a particular source for the probability of detection based on evaluating it with an available baseline algorithm. From these model estimate, target probabilities of detection were specified for each of the training, public test and private test regions, and the weights used by the NUSF methodology were adjusted. Larger weights are emphasized more, with larger values leading to greater concentrations of design points.

For each of the training, public test and private test sets, a region of emphasis was identified, and then designs were created with the appropriate sample size and density of points for each of the data sets. This process was repeated for each source with and without shielding. Figure 62 shows the three designs generated using the NUSF approach for a given source without shielding. The yellow points represent the input parameter combinations for the training data set. The purple points are those design locations for the public test set. The blue points are those for the private test set. As noted, the goal of the design was to encourage and reward the development of algorithms that were able to adapt and perform well for new more challenging scenarios. This approach also penalizes competitors who overfit their solutions based on feedback from the public leaderboard.

The new methodology provided more automated design construction capability and made it easier to construct the large volumes and sizes of designs from massive supersets of candidate points.

In addition to the data competition example, we propose other situations where it can be desirable to control the degree of concentration of data points in various regions, while still maintaining some space-filling properties:

1. Designs with the goal of optimizing a response, while still maintaining the ability to explore throughout the region. For example, if historical data suggest that a process gives improved yield for some factor combinations, the goal for an experiment might be to place additional runs near this optimum by overemphasizing this region, while still allowing for possible new regions of good performance to be identified throughout the region. There can be advantages of improved overall estimation of a parametric model if data are gathered throughout the region of interest.

2. Focus on interesting features. For example, consider a chemical process with a phase change in the input region of interest. In this case, it may be desirable to over-represent locations close to the phase change to gain additional precision of estimation in this region, where rapid changes in the response are anticipated.

3. Focus on regions with larger variability. If historical data are available for a process, there may be regions known to be less precisely estimated based on current information, because of model uncertainty or sparsity of previous data. In these cases, it can be beneficial to emphasize new data collection in these regions, while still enhancing overall estimation throughout the region.

4. Focus on regions where the function is changing rapidly. In some processes, there are known regions where the "wiggliness" or complexity of the function is known to be much more prevalent than in other regions. To improve precision, it is advantageous to place more data in these regions than in areas where the surface changes more slowly.

5. Focus on regions with greater discrepancy between the computer model and observed data. In calibration experiments, data are often collected to determine the degree of matching between a science model of the process and what is observed in nature. If historical data suggest regions with less agreement between these two sources of information, over-representing these regions can be helpful to further understand these inconsistencies. For example, a higher density of design points (corresponding to

larger weights) may be desired in a region that has shown a larger discrepancy between the actual observation and the estimate from a fitted model. The flexibility of the NUSF approach is that it allows the practitioner to specify whatever structure is desired for the weights to achieve changes in emphasis.

To construct a design, the following process is recommended [5]: The basic steps of the process are as follows:

1. Select a candidate set of points from which the design will be constructed.
2. Identify preliminary weights associated with each candidate point to match the pattern of the desired density across the input space.
3. Scale the preliminary weights into one (or more ranges) of scaled weights in the interval [1, MWR], where MWR is the maximum weight ratio which is a user specified value greater than 1, which controls the degree of non-uniformity. Note the scaled weights will be used to calculate the weighted distance in the nonuniform design space. A larger MWR value results in the design points being more packed in the higher density region. However, the impact of MWR value is affected by both the distribution of the weight values in the design region and the number of design points.
4. Since the experimenter may only be able to judge whether the degree of nonuniformity they seek has been achieved by considering several candidates, generate several NUSF designs based on different MWR values.
5. Compare alternative designs to select one that best matches the design objectives.

This process was used in the construction of all of the TopCoder competition data sets generated.

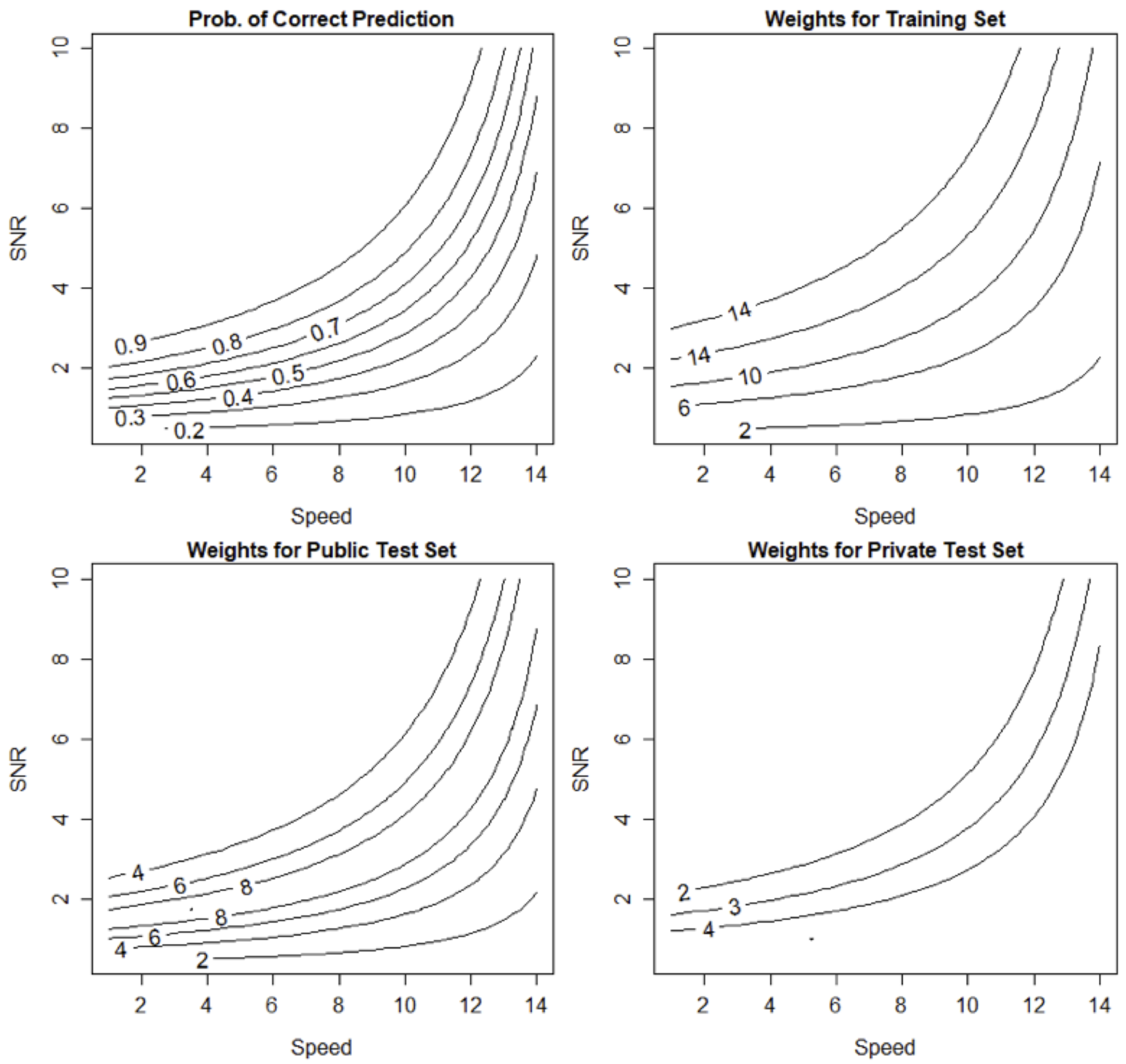


Figure 61: Example of estimated model for estimating probability of detection for one source without shielding. The weights for the NUSF were constructed by emphasizing particular probabilities for each of the training, public test and private test sets.

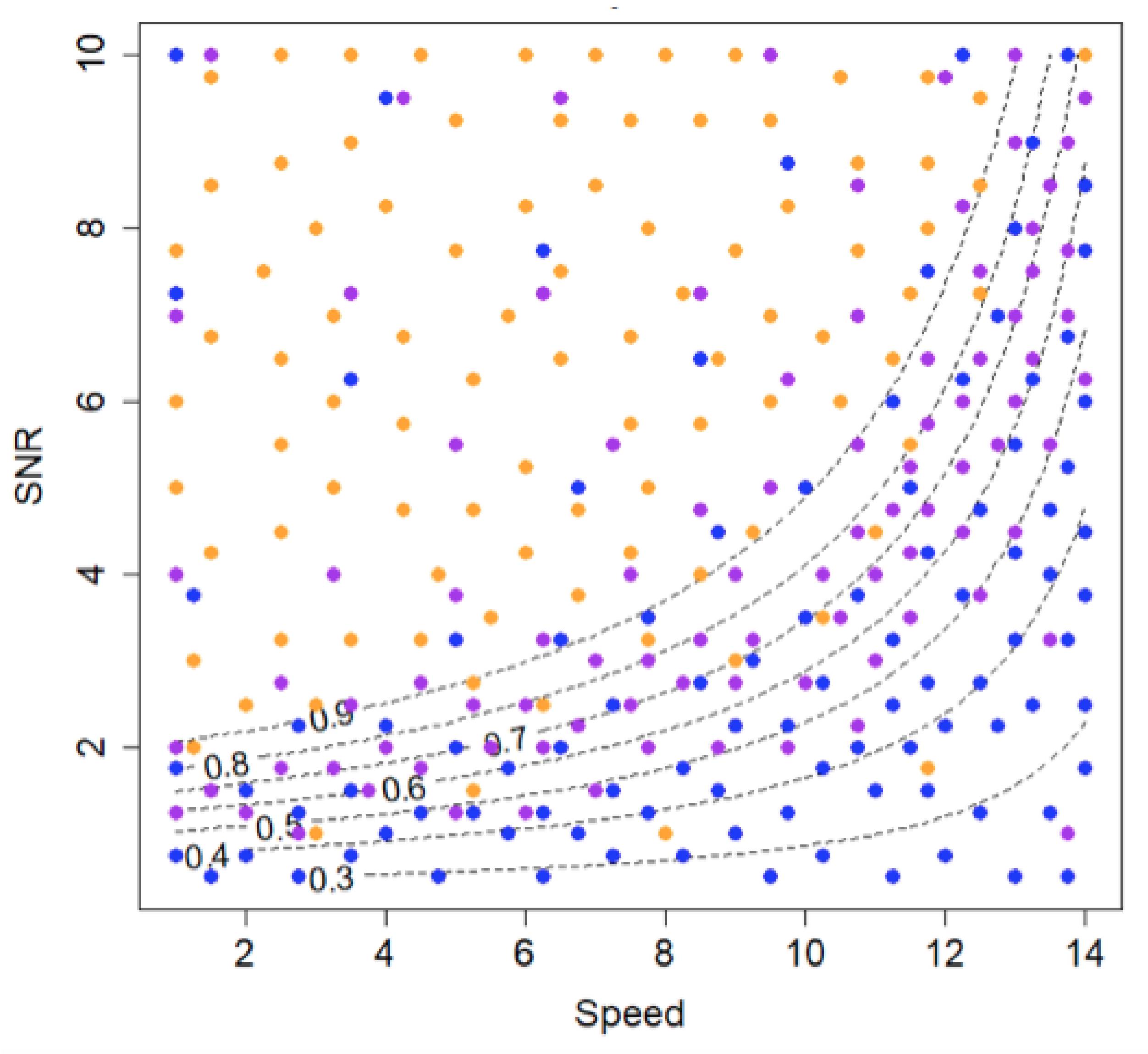


Figure 62: Sample training, public test and private test set designs used for a particular source without shielding.

## 6.3 Comparing Algorithms with Uncertainty in Leaderboard Position

In this section, we describe a simple transparent strategy for understanding the robustness of the algorithm performance to potential uncertainty of the competition data and propose a new strategy for more equitably distributing the prize that reflects the close proximity of many of the final scores [96]. This new strategy preserves rewarding the competition priorities established by the host, while still taking into account the inherent uncertainty that is known to affect the exact value of the score. There are four ways to use the proposed bootstrap methodology:

1. During the competition, we propose providing the competitors with a summary of the distribution of their rankings across multiple bootstrap resamples of the public test set data. This will show them how often they were in different prize positions, and encourage both those who are ahead and behind to work to further improve their algorithms.
2. After the competition, we recommend using the bootstrap resampling to determine the allocation of prize money. This will allow competitors who finished with very similar scores to be rewarded based on the proportion of time that they were in each ranked position.
3. For the competition host, we provide numerical and graphical summaries to help them understand which competitor results might reasonably considered "effectively the same", and which are "notably different".
4. Finally, the flexibility of the resampling approach allows the competition host to explore "what if" scenarios to see how rankings would change if some parts of the data were emphasized or de-emphasized, or if the leaderboard scoring formula was adjusted.

The main advantages of the new approach are:

- to maintain a fixed total purse for prizes, but with flexibility for how they are distributed according to the algorithm performance across a variety of possible data,
- to offer greater fairness to the competitors, which should be attractive to encourage broad and diverse participation,
- to provide more informative public leaderboard feedback to reward solutions with better generalization abilities and to help prevent overfitting to the particular set of competition data, and
- to give an opportunity after the competition is over to easily examine how results would have changed under different scenarios with altered interests or competition goals.

The key idea of the new approach is to use stratified standard or fractionally weighted bootstrap sampling to quantify the uncertainty associated with the scores for each competitor. This is done by obtaining data sets that share the priorities specified by the host, but allows us to assess the uncertainty of the leaderboard scores. For example, in the urban radiation search competition, the data sets were comprised of runs. The relevant runs for scoring were resampled maintaining the balance of runs desired by the host, and new scores and rankings were obtained for each new data set.

The methodology was used on the TopCoder data to understand the overlap in the competitor scores. Figure 63 shows the range of scores from the private leaderboard for each competitor as a cumulative distribution function (CDF). The range of scores achieved by each competitor provides information about how much uncertainty there was in estimating their score based on a single data set. If curves for competitors are in close proximity, this suggests that their relative position in the competition might change with a different data set. A prominent pattern of this figure is the clustering of some of the CDF curves. It is apparent that the top team is well separated with the 2nd and the 3rd teams with a substantially higher private test scores across all the resampled data. Between the 2nd and 3rd place teams, even though the 2nd place team seems to slightly outperform the 3rd place team, their CDF curves are very closely located with considerable overlap in the ranges of the observed scores. Similar pattern can be observed for the grouping of Teams 4-8, whose CDF curves are clustered with very similar performance ranging between 71-72 points. The group of next tier of performance include the 9th and 10th place teams those scores generally range between 69-70 points. The 11th place team performs generally about 0.6-1 points lower than the 9th and 10th place teams. The 12th place team is not comparable with the top 10 teams with scores that are generally at least 4 points below the 10th place team.

Figure 64 shows how often different competitors ended up ranked in the various locations. The summary serves to emphasize how dominant the 1st place competitor was as he was never any lower than 1st place for any resample. Even though the performance of 2nd and 3rd place competitors are similar, there is still quite a noticeable difference between the two competitors with the 2nd place competitor outperforming the 3rd place competitor more than 85% of all the resamples. Among competitors 4-8, the 4th and 5th place competitors generally outperform competitors 6-8 with the latter having closer performance among one another. The 4th place competitor finished in 4th for more than 60% of the resamples, while the 5th place competitor finished in 4th approximately 35% of the resamples. Competitors 6, 7 and 8 each finish in 4th a small (but non-zero) fraction of the time.

The ranking of competitors 9-10 are reversed for about 17% of the possible resamples while the 9th place competitor still outperforms 10th place competitor 83% of the time. The 11th place competitor is rarely among the top 10 except for a very small fraction of possible scenarios (in about 5.9% cases he ranked as the 10th place). The 12th place competitor is never ranked among the top 10 teams across all simulations.

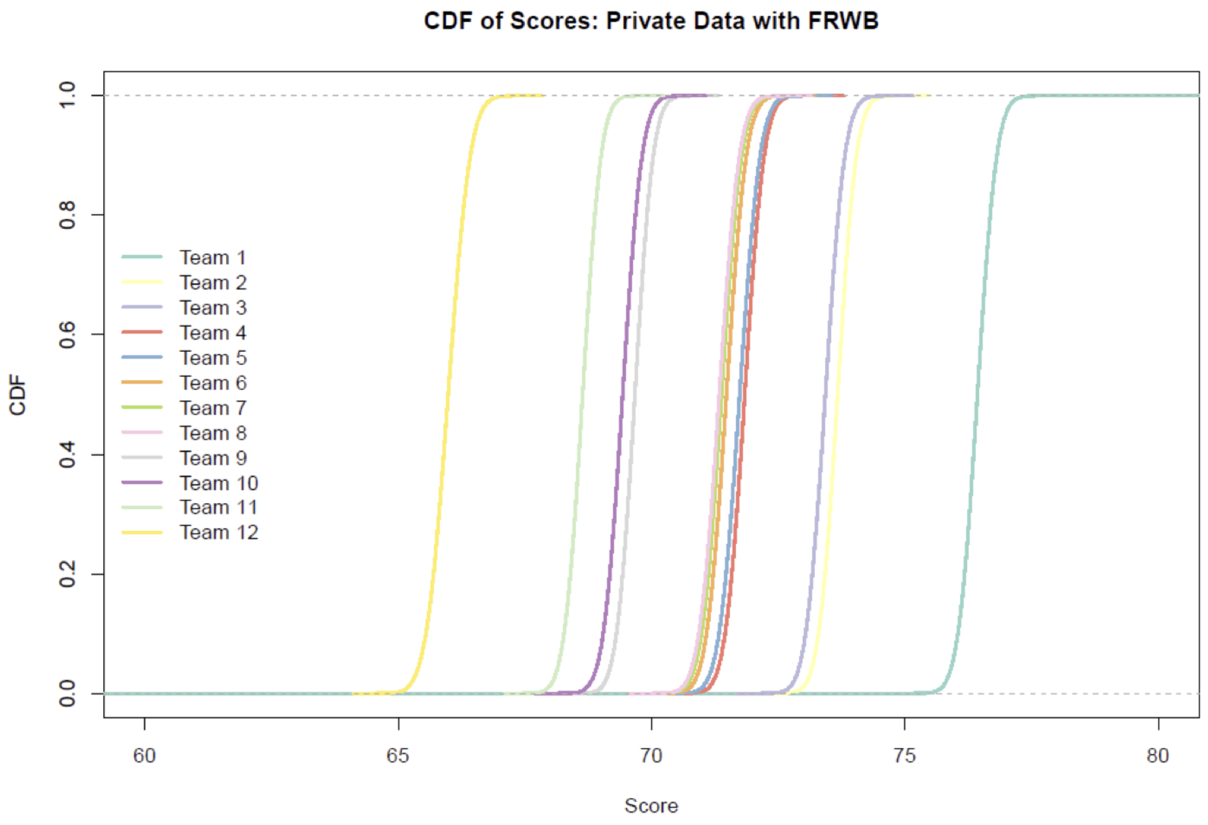


Figure 63: Distribution of competitor private leaderboard scores across similar datasets that preserve the emphasis on different sources.

Given the considerable differences in the prize money for each place in the competition, we provide a new prize distribution strategy that more directly considers the robustness of algorithm performance to different possible data and offers a fairer allocation of the prize money that reflects the different places achieved by the competitors in the many resamples of the data. The idea is to distribute the total prize money for all rankings evenly across the large number of bootstrap resamples, then allocate the designated prize money based on the ranking of the teams for each resample, and finally add up the allocated "per resample prize

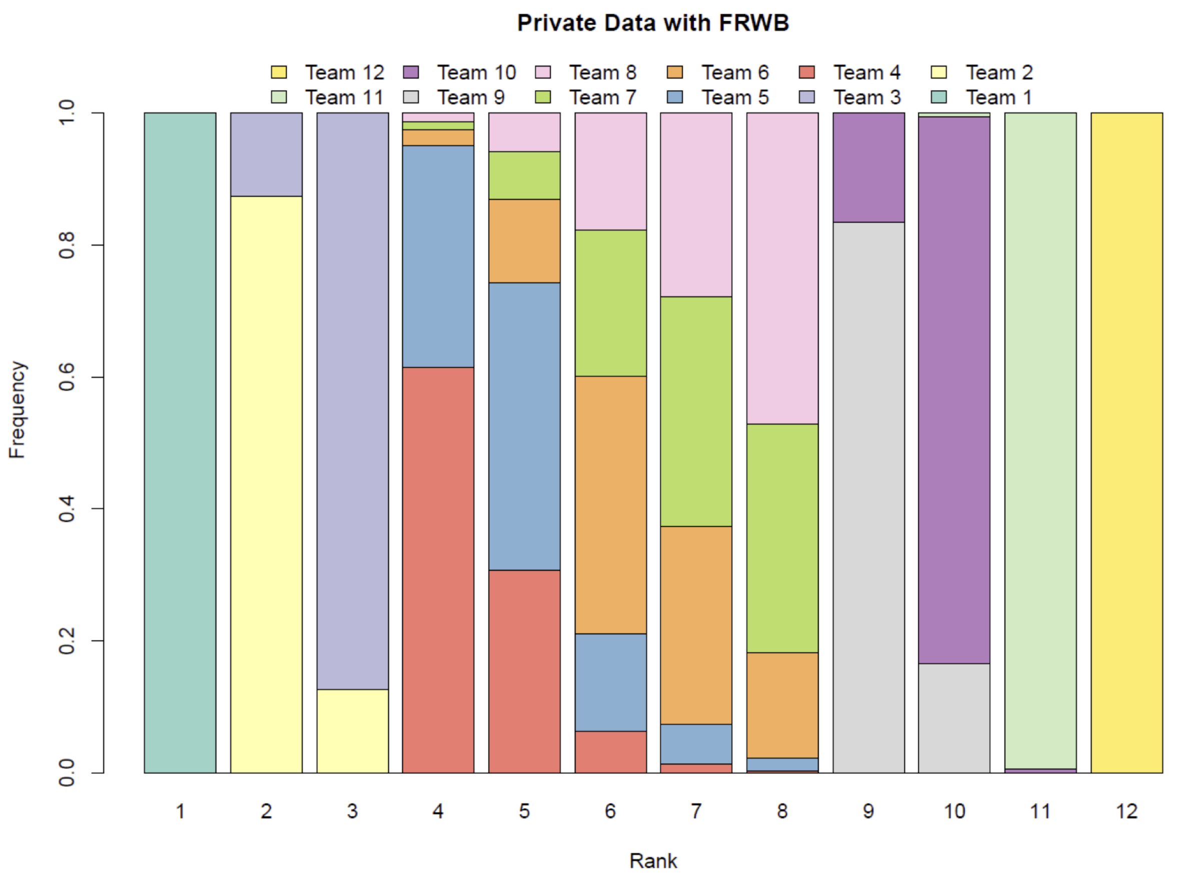
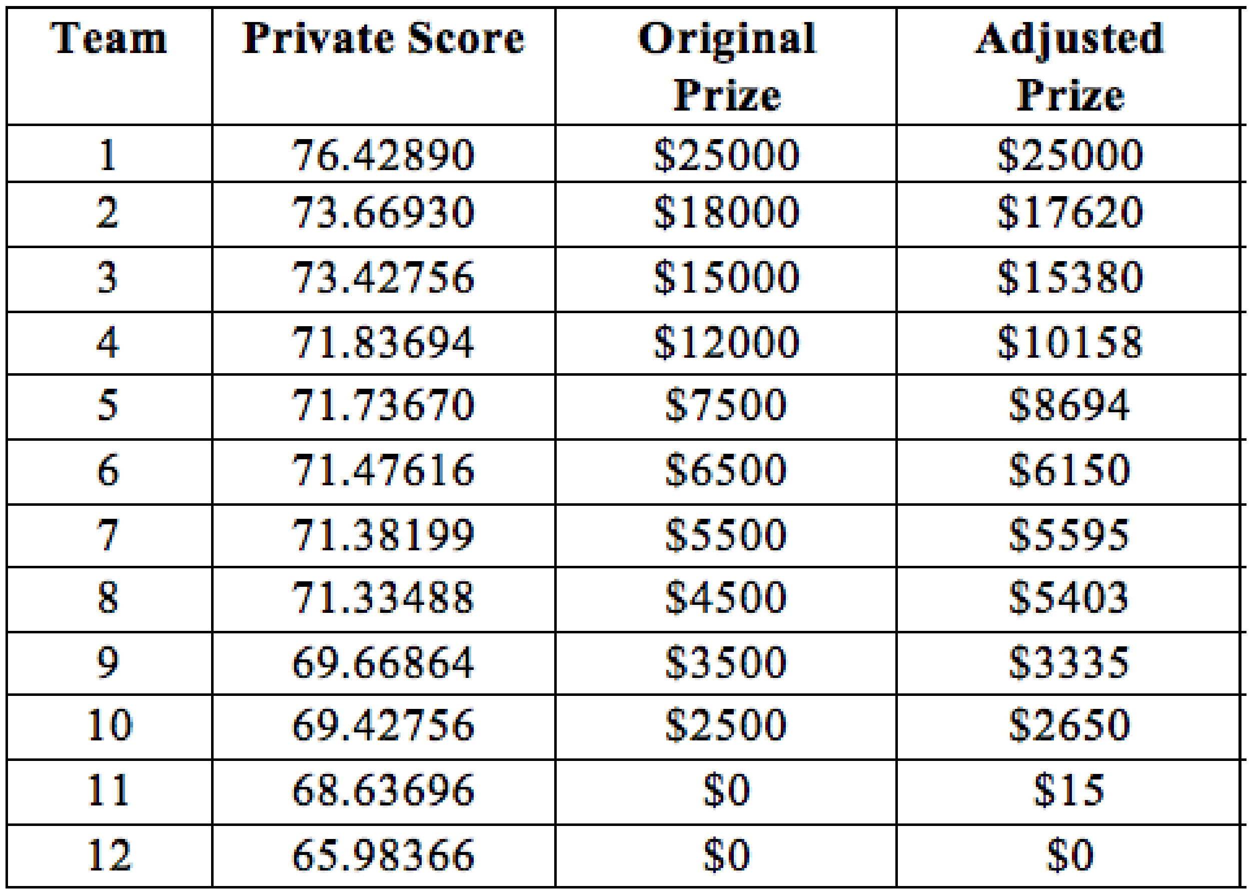


Figure 64: Summary of the different rankings achieved by the top competitors across the bootstrapped similar data sets.

money" across all the resamples to obtain the total adjusted prize for each team. This strategy ensures a fixed total purse of money and allows flexibility to distribute the money based on a fairer evaluation across many possible data from the general problem.

Figure 65 shows a summary table of the hypothetical prize allocation if the suggested methodology had been used. compares the original prize and the adjusted prize based on the proposed new strategy. Because of the dominance of the competition winner in each of the resamples, we see that the leading competitor still receives the exact total amount at $25,000 since he was never in the 2nd place across all simulations. Since the 2nd and 3rd place competitors switched their rankings for about 15% of the scenarios, the adjusted prize for the 2nd place competitor has been reduced for $380, which is awarded to the 3rd place competitor. Hence there is a proportionate blending of the two prize amounts to reflect that competitor 2 does not always dominate competitor 3. Similarly, the prize for the 4th to 8th place competitors have been rebalanced with the 5th, 7th and 8th place competitors' prizes increasing by $1194, $95, and $903, respectively to the detriment of competitor 4, to reflect the proportion of times that each team spent in each of the ranks. The prize for the 9th place competitor was reduced by $165, of which $150 has been reallocated to the 10th competitor and surprisingly $15 would have been given to the 11th place competitor, who originally did not receive any prize money, but occasionally appears among the top ten in the simulations.

The goal of the new prize allocation is to capture the intermingling of the ranks for the different teams across the resampled data sets, and to temper the extreme differences between the prize amounts between places when teams have similar performance.



| Team | Private Score | Original Prize | Adjusted Prize |
|---|---|---|---|
| 1 | 76.42890 | $25000 | $25000 |
| 2 | 73.66930 | $18000 | $17620 |
| 3 | 73.42756 | $15000 | $15380 |
| 4 | 71.83694 | $12000 | $10158 |
| 5 | 71.73670 | $7500 | $8694 |
| 6 | 71.47616 | $6500 | $6150 |
| 7 | 71.38199 | $5500 | $5595 |
| 8 | 71.33488 | $4500 | $5403 |
| 9 | 69.66864 | $3500 | $3335 |
| 10 | 69.42756 | $2500 | $2650 |
| 11 | 68.63696 | $0 | $15 |
| 12 | 65.98366 | $0 | $0 |

Figure 65: Hypothetical prize redistribution under the resampling methodology proposed.

## 6.4 Pareto Fronts for Assessing Subsets of Competition Criteria

In many data competitions, prizes are awarded for excellent performance across a number of different objectives. In order to rank and make direct comparisons between competing solutions, the scores need to be a scalar that combines the different aspects of the competition into a single number. For the urban radiation search competition, multiple different objectives were simultaneously considered using simulated data.

To create the leaderboard scoring formula, each of these objectives was first quantified and then combined. Considerable thought and planning went into developing the right balance of these objectives as this is known to have an important effect on the priorities that the competitors place on improving different components of their algorithm. Regardless of how well the host formulates the leaderboard, it represents just a single way of combining the different objectives into an overall score. This predetermined formula drives the development of competition solutions towards the intended balance between the multiple objects of interest. This setup is typical of many data competitions where multiple objectives, with potentially different importance, are assessed through the available data presented to the competitors. Being able to uncouple the contributions of each objective and evaluate the ability of each solution to perform are important for understanding the merits of the available alternatives.

When the competition concludes, the host generally gains access to the winning solutions with the goal of implementing them for use in new scenarios. To take advantage of the financial and effort investment in the competition, the host would like to be able to select the best solution for variations of the problem that fall within the scope of the competition. This might involve only a subset of the objectives and might prefer a different emphasis on the objectives than was originally captured in the leaderboard scoring formula.

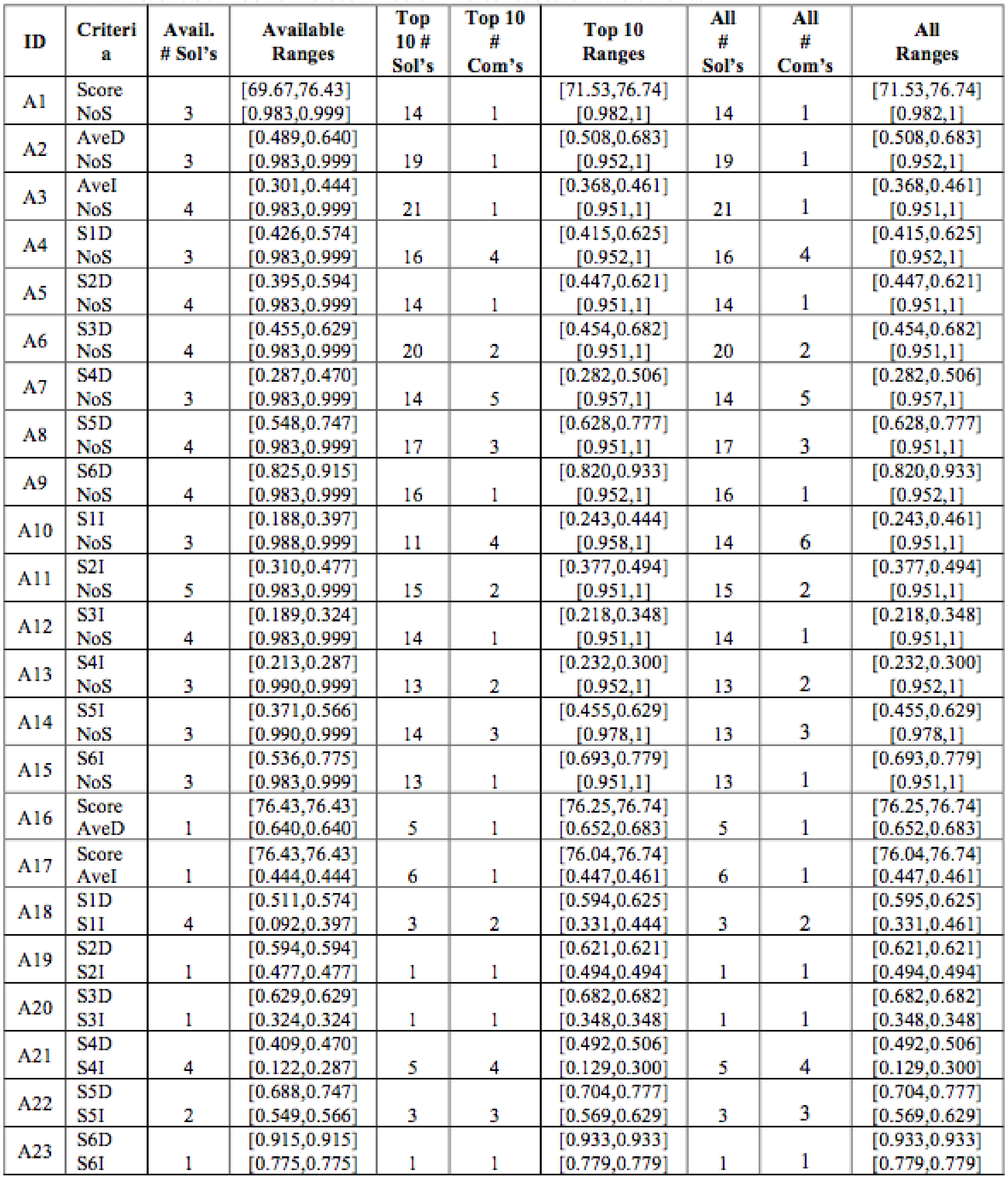
Here we outline an approach to understanding the relative merits of different competitor algorithms, and provides a path to select the most appropriate one for implementation in a new scenario. The approach is general enough to allow different subsets of objectives to be the focus, with the best solutions identified for a variety of emphases of the objectives [97].

Once the competition was completed, a set of summarizing criteria was identified in the post-competition analysis to evaluate the competitor's performance on different objectives of detection and identification across different types of sources and no source scenarios. Analysis revealed there was little to no difference between competitors on the locate aspect – if they were able to detect or identify a source in the run, then they were able to correctly locate it. Hence, it was not considered to be important to distinguish between the different solutions. This is an important strategy to reduce the dimensionality of the problem. If some criteria do not offer any practical distinction between top solutions, they can be removed from further consideration. The objective metrics considered for different scenarios of interest include:

1. S1D,... S6D – proportion of each of the Source runs correctly detected. Range [0,1]. Maximize.
2. S1, ... S6I – proportion of each of the Source runs correctly identified. Range [0,1]. Maximize
3. NoS – proportion of no source runs correctly labeled (FPR = 1 – NoS). Range [0,1]. Maximize.
4. AveD – average proportion of sources detected (over all 6 sources). Range [0,1]. Maximize.
5. AveI - average proportion of sources correctly identified (over all 6 sources). Range [0,1]. Maximize.
6. Score – private test score (considered a proxy for robust overall performance). Range [0,100]. Maximize.

Pareto fronts [98] [99] [100] can objectively eliminate non-competitive, or dominated solutions. One solution is said to dominate another if all of its criteria values are at least as good as those of another solution with at least one of its criterion value being strictly better. For example, if we seek to maximize both S1D and S1I, one algorithm dominates another if the former has both S1D and S1I at least as high as the latter, and at least one of S1D or S1I is strictly higher. After this objective stage, determining the best solution for a particular balancing of the criteria involves using some graphical tools to compare between the alternative Pareto front (PF) solutions to identify one that works best for the subjective prioritization of the scenario.

We now consider multiple possible scenarios identified as being of potential interest. For many data competitions, it is likely that all of the objectives in the leaderboard scoring formula are not of equal importance and may not be relevant for all problems under consideration. It is helpful to define the sets of subsets of useful combinations of the objectives and explore the performance of the competitors' solutions for each of them. This helps to provide insights into the diversity of solutions that have been obtained, and gain understanding about the strengths and weaknesses of each approach. Since having a high false positive rate (FPR) is highly undesirable for this competition, we eliminate all of the solutions that have a FPR greater than 5% from further consideration.



| ID | Criteri a | Avail. # Sol's | Available Ranges | Top 10 # Sol's | Top 10 # Com's | Top 10 Ranges | All # Sol's | All # Com's | All Ranges |
|---|---|---|---|---|---|---|---|---|---|
| A1 | Score<br>NoS | 3 | [69.67,76.43]<br>[0.983,0.999] | 14 | 1 | [71.53,76.74]<br>[0.982,1] | 14 | 1 | [71.53,76.74]<br>[0.982,1] |
| A2 | AveD<br>NoS | 3 | [0.489,0.640]<br>[0.983,0.999] | 19 | 1 | [0.508,0.683]<br>[0.952,1] | 19 | 1 | [0.508,0.683]<br>[0.952,1] |
| A3 | AveI<br>NoS | 4 | [0.301,0.444]<br>[0.983,0.999] | 21 | 1 | [0.368,0.461]<br>[0.951,1] | 21 | 1 | [0.368,0.461]<br>[0.951,1] |
| A4 | S1D<br>NoS | 3 | [0.426,0.574]<br>[0.983,0.999] | 16 | 4 | [0.415,0.625]<br>[0.952,1] | 16 | 4 | [0.415,0.625]<br>[0.952,1] |
| A5 | S2D<br>NoS | 4 | [0.395,0.594]<br>[0.983,0.999] | 14 | 1 | [0.447,0.621]<br>[0.951,1] | 14 | 1 | [0.447,0.621]<br>[0.951,1] |
| A6 | S3D<br>NoS | 4 | [0.455,0.629]<br>[0.983,0.999] | 20 | 2 | [0.454,0.682]<br>[0.951,1] | 20 | 2 | [0.454,0.682]<br>[0.951,1] |
| A7 | S4D<br>NoS | 3 | [0.287,0.470]<br>[0.983,0.999] | 14 | 5 | [0.282,0.506]<br>[0.957,1] | 14 | 5 | [0.282,0.506]<br>[0.957,1] |
| A8 | S5D<br>NoS | 4 | [0.548,0.747]<br>[0.983,0.999] | 17 | 3 | [0.628,0.777]<br>[0.951,1] | 17 | 3 | [0.628,0.777]<br>[0.951,1] |
| A9 | S6D<br>NoS | 4 | [0.825,0.915]<br>[0.983,0.999] | 16 | 1 | [0.820,0.933]<br>[0.952,1] | 16 | 1 | [0.820,0.933]<br>[0.952,1] |
| A10 | S1I<br>NoS | 3 | [0.188,0.397]<br>[0.988,0.999] | 11 | 4 | [0.243,0.444]<br>[0.958,1] | 14 | 6 | [0.243,0.461]<br>[0.951,1] |
| A11 | S2I<br>NoS | 5 | [0.310,0.477]<br>[0.983,0.999] | 15 | 2 | [0.377,0.494]<br>[0.951,1] | 15 | 2 | [0.377,0.494]<br>[0.951,1] |
| A12 | S3I<br>NoS | 4 | [0.189,0.324]<br>[0.983,0.999] | 14 | 1 | [0.218,0.348]<br>[0.951,1] | 14 | 1 | [0.218,0.348]<br>[0.951,1] |
| A13 | S4I<br>NoS | 3 | [0.213,0.287]<br>[0.990,0.999] | 13 | 2 | [0.232,0.300]<br>[0.952,1] | 13 | 2 | [0.232,0.300]<br>[0.952,1] |
| A14 | S5I<br>NoS | 3 | [0.371,0.566]<br>[0.990,0.999] | 14 | 3 | [0.455,0.629]<br>[0.978,1] | 13 | 3 | [0.455,0.629]<br>[0.978,1] |
| A15 | S6I<br>NoS | 3 | [0.536,0.775]<br>[0.983,0.999] | 13 | 1 | [0.693,0.779]<br>[0.951,1] | 13 | 1 | [0.693,0.779]<br>[0.951,1] |
| A16 | Score<br>AveD | 1 | [76.43,76.43]<br>[0.640,0.640] | 5 | 1 | [76.25,76.74]<br>[0.652,0.683] | 5 | 1 | [76.25,76.74]<br>[0.652,0.683] |
| A17 | Score<br>AveI | 1 | [76.43,76.43]<br>[0.444,0.444] | 6 | 1 | [76.04,76.74]<br>[0.447,0.461] | 6 | 1 | [76.04,76.74]<br>[0.447,0.461] |
| A18 | S1D<br>S1I | 4 | [0.511,0.574]<br>[0.092,0.397] | 3 | 2 | [0.594,0.625]<br>[0.331,0.444] | 3 | 2 | [0.595,0.625]<br>[0.331,0.461] |
| A19 | S2D<br>S2I | 1 | [0.594,0.594]<br>[0.477,0.477] | 1 | 1 | [0.621,0.621]<br>[0.494,0.494] | 1 | 1 | [0.621,0.621]<br>[0.494,0.494] |
| A20 | S3D<br>S3I | 1 | [0.629,0.629]<br>[0.324,0.324] | 1 | 1 | [0.682,0.682]<br>[0.348,0.348] | 1 | 1 | [0.682,0.682]<br>[0.348,0.348] |
| A21 | S4D<br>S4I | 4 | [0.409,0.470]<br>[0.122,0.287] | 5 | 4 | [0.492,0.506]<br>[0.129,0.300] | 5 | 4 | [0.492,0.506]<br>[0.129,0.300] |
| A22 | S5D<br>S5I | 2 | [0.688,0.747]<br>[0.549,0.566] | 3 | 3 | [0.704,0.777]<br>[0.569,0.629] | 3 | 3 | [0.704,0.777]<br>[0.569,0.629] |
| A23 | S6D<br>S6I | 1 | [0.915,0.915]<br>[0.775,0.775] | 1 | 1 | [0.933,0.933]<br>[0.779,0.779] | 1 | 1 | [0.933,0.933]<br>[0.779,0.779] |

Figure 66: Results for 23 2-criterion Pareto fronts, based on different collections of the solutions.

Figure 66 shows a table of 23 2-criteria scenarios of potential interest to the experts. As you can see, not all 16 choose 2 combinations of the criteria are of interest, but the scenarios identified consider sensible pairings of criteria that might arise for possible implementation. In selecting the subset of scenarios, score was treated as a proxy for good robust overall performance across all 16 criteria. Many of the combinations consider a low false positive rate (or large NoS value) in combination with one other criterion. This reflects the emphasis on not having too many false alarms to hinder effective implementation. Other pairs of criteria focus on a single source, which might reflect a scenario with the goal of tracking that source.

There are three Pareto fronts described for each scenario. The “Available” column considers the final submission from each of the top 10 prize winning competitors. The “Top 10” columns describe the results based on all 604 submissions from any of the prizewinners. Finally, the ”All” columns include all of he 1037 submissions from all competitors. There are columns for the number of solutions on the Pareto front, as well as the number of competitors with at least one submission on the front.

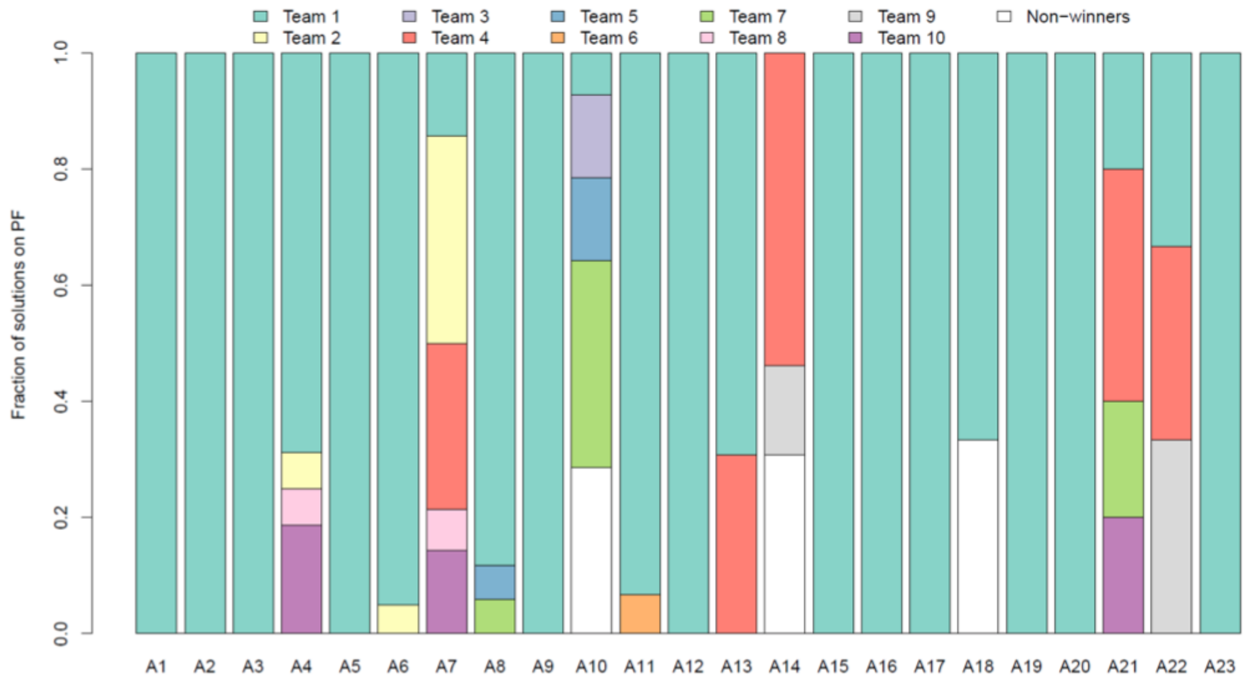


Figure 67: Barplot of the representation of different competitors on the Pareto front for each of the 2-criterion scenarios.

Figure 67 shows the frequency with which different competitors appear on the Pareto front for the different scenarios described in Figure 66.

The first place finisher’s solutions appear on the PFs for 22 out of the 23 scenarios. The only scenario for which the top winner’s solution is not included on the PF occurs when the primary focus is for detecting source 4 while controlling the FPR (scenario A14). Also, for 12 out of the 23 scenarios, only solutions from the winner, pfr, appear on the PF. Most of these scenarios focus on detecting source 2 or 6, identifying source 3, or using the overall score or the average detection or identification rate across all sources, while balancing the FPR. On the other hand, when the focus is on other sources, other prizewinners become more competitive. For example, the 2nd place finisher does well for detecting sources 1, 3, and 4. The 8th and 10th place finishers are also competitive for detecting sources 1 and 4. The 4th place finisher does quite well on identifying sources 4 & 5. And the 5th and 7th place finishers seem competitive at detecting source 5 and identifying source 1.

| ID | Criteria | # Final Sub's | Available Ranges | Top 10 # Sol's | Top 10 # Com's | Top 10 Ranges | All # Sol's | All # Com's | All Ranges |
|---|---|---|---|---|---|---|---|---|---|
| B1 | AveD<br>AveI<br>NoS | 4 | [0.489,0.640]<br>[0.301,0.444]<br>[0.983,0.999] | 23 | 1 | [0.508,0.683]<br>[0.368,0.461]<br>[0.951,1] | 23 | 1 | [0.508,0.683]<br>[0.368,0.461]<br>[0.951,1] |
| B2 | Score<br>AveD<br>NoS | 3 | [69.67,76.43]<br>[0.489,0.641]<br>[0.983,0.999] | 20 | 1 | [71.53,76.74]<br>[0.508,0.683]<br>[0.952,1] | 20 | 1 | [71.53,76.74]<br>[0.508,0.683]<br>[0.952,1] |
| B3 | Score<br>AveI<br>NoS | 5 | [69.67,76.43]<br>[0.301,0.444]<br>[0.983,0.999] | 22 | 1 | [71.53,76.74]<br>[0.368,0.461]<br>[0.951,1] | 22 | 1 | [71.53,76.74]<br>[0.368,0.461]<br>[0.951,1] |
| B4 | S1D<br>S1I<br>NoS | 6 | [0.426,0.574]<br>[0.092,0.397]<br>[0.976,0.999] | 58 | 7 | [0.415,0.625]<br>[0.084,0.444]<br>[0.952,1] | 62 | 9 | [0.391,0.625]<br>[0.084,0.461]<br>[0.951,1] |
| B5 | S2D<br>S2I<br>NoS | 5 | [0.395,0.594]<br>[0.310,0.477]<br>[0.983,0.999] | 15 | 2 | [0.447,0.621]<br>[0.377,0.494]<br>[0.951,1] | 15 | 2 | [0.447,0.621]<br>[0.377,0.494]<br>[0.951,1] |
| B6 | S3D<br>S3I<br>NoS | 4 | [0.455,0.629]<br>[0.189,0.324]<br>[0.983,0.999] | 21 | 3 | [0.454,0.682]<br>[0.204,0.348]<br>[0.951,1] | 21 | 3 | [0.454,0.682]<br>[0.204,0.348]<br>[0.951,1] |
| B7 | S4D<br>S4I<br>NoS | 5 | [0.287,0.470]<br>[0.122,0.287]<br>[0.976,0.999] | 42 | 6 | [0.282,0.506]<br>[0.099,0.300]<br>[0.952,1] | 42 | 6 | [0.282,0.506]<br>[0.099,0.300]<br>[0.952,1] |
| B8 | S5D<br>S5I<br>NoS | 4 | [0.548,0.747]<br>[0.371,0.567]<br>[0.983,0.999] | 35 | 5 | [0.532,0.777]<br>[0.445,0.629]<br>[0.951,1] | 37 | 6 | [0.532,0.777]<br>[0.445,0.629]<br>[0.951,1] |
| B9 | S6D<br>S6I<br>NoS | 4 | [0.825,0.915]<br>[0.221,0.775]<br>[0.983,0.999] | 25 | 1 | [0.820,0.933]<br>[0.693,0.779]<br>[0.951,1] | 25 | 1 | [0.820,0.933]<br>[0.693,0.779]<br>[0.951,1] |
| B10 | Score<br>AveD<br>AveI | 1 | [76.43,76.43]<br>[0.640,0.640]<br>[0.444,0.444] | 6 | 1 | [76.04,76.74]<br>[0.652,0.683]<br>[0.447,0.461] | 6 | 1 | [76.04,76.74]<br>[0.652,0.683]<br>[0.447,0.461] |
| B11 | Score<br>S1D<br>S1I | 5 | [68.64,76.43]<br>[0.511,0.574]<br>[0.092,0.397] | 11 | 4 | [71.84,76.74]<br>[0.526,0.625]<br>[0.308,0.444] | 11 | 5 | [71.84,76.74]<br>[0.526,0.625]<br>[0.308,0.461] |
| B12 | Score<br>S2D<br>S2I | 1 | [76.43,76.43]<br>[0.594,0.594]<br>[0.477,0.477] | 8 | 1 | [76.04,76.74]<br>[0.581,0.621]<br>[0.464,0.494] | 8 | 1 | [76.04,76.74]<br>[0.581,0.621]<br>[0.464,0.494] |
| B13 | Score<br>S3D<br>S3I | 1 | [76.43,76.43]<br>[0.629,0.629]<br>[0.324,0.324] | 8 | 1 | [76.04,76.74]<br>[0.637,0.682]<br>[0.316,0.348] | 8 | 1 | [76.04,76.74]<br>[0.637,0.682]<br>[0.316,0.348] |
| B14 | Score<br>S4D<br>S4I | 5 | [68.64,76.43]<br>[0.409,0.470]<br>[0.122,0.287] | 10 | 4 | [68.62,76.74]<br>[0.452,0.506]<br>[0.129,0.300] | 10 | 4 | [68.62,76.74]<br>[0.452,0.506]<br>[0.129,0.300] |
| B15 | Score<br>S5D<br>S5I | 2 | [71.84,76.43]<br>[0.688,0.747]<br>[0.549,0.567] | 11 | 3 | [69.01,76.74]<br>[0.689,0.777]<br>[0.534,0.629] | 11 | 3 | [69.01,76.74]<br>[0.689,0.777]<br>[0.534,0.629] |
| B16 | Score<br>S6D<br>S6I | 1 | [76.43,76.43]<br>[0.915,0.915]<br>[0.775,0.775] | 9 | 1 | [76.04,76.74]<br>[0.919,0.933]<br>[0.765,0.779] | 9 | 1 | [76.04,76.74]<br>[0.919,0.933]<br>[0.765,0.779] |

Figure 68: Results for 16 3-criterion Pareto fronts, based on different collections of the solutions.

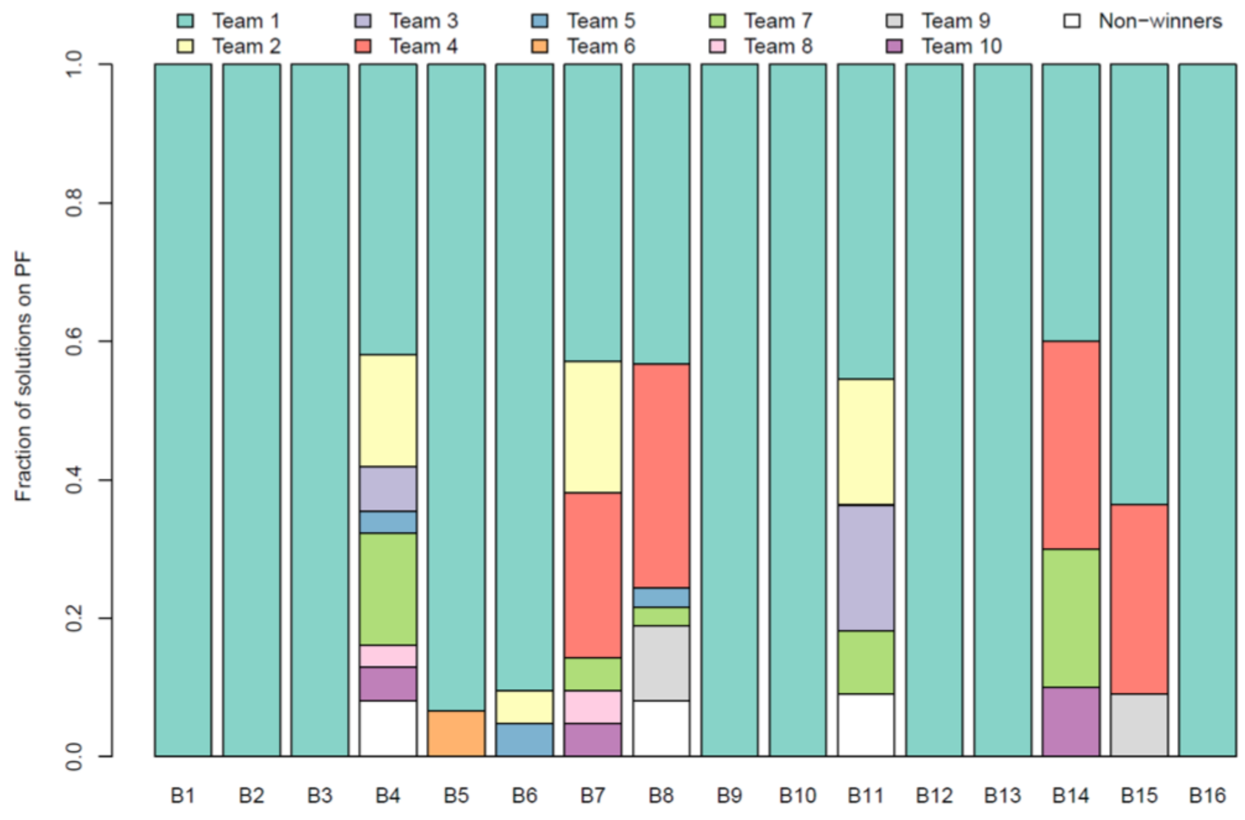


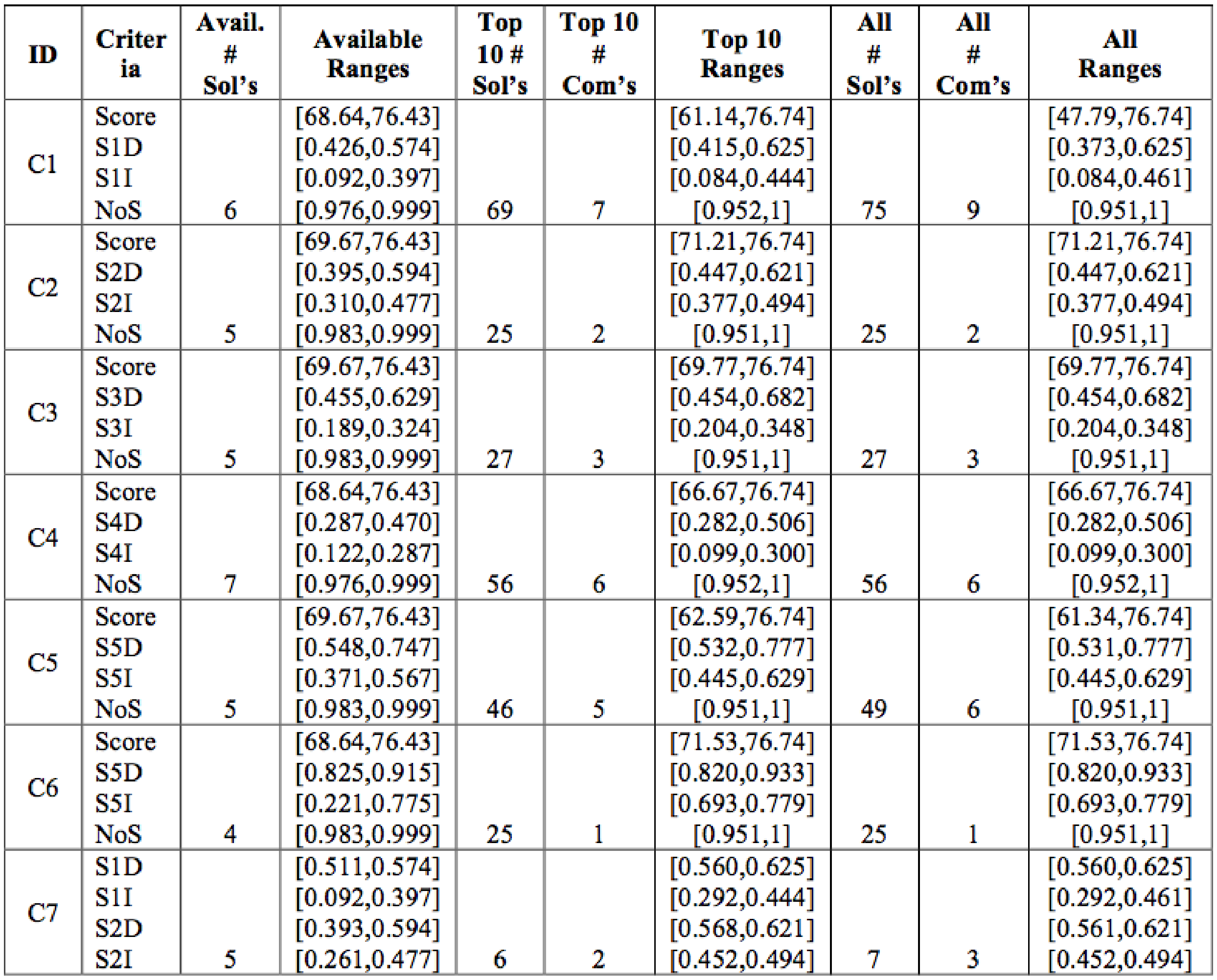
Figure 69: Barplot of the representation of different competitors on the Pareto front for each of the 3-criterion scenarios.

Other scenarios involving 3 criteria are shown in Figures 68 and 69. We see that the prizewinners' solutions still generally dominate most of the competitive choices on the PFs across all the scenarios of interest. For half of the cases, the first place finisher has all non-dominated solutions for situations focusing on the overall score, average performance across all sources, or individual scenarios concerning identifying sources 2, 3, or 6. In contrast, when source 1, 4, or 5 is of interest, other prizewinners become competitive. This confirms that the leaderboard scoring formula did a great job of selecting the best solutions across a large number of objectives and scenarios. However, when it comes to a particular implementation scenario, more tailored solutions can be identified by carefully examining the merits of solutions for a subset of important objectives.

We also consider some scenarios with larger numbers of criteria. The tables in Figures 70 and 71 describe some higher dimensional Pareto fronts. The accompanying barplots for these scenarios are shown in Figures 72 and 73.

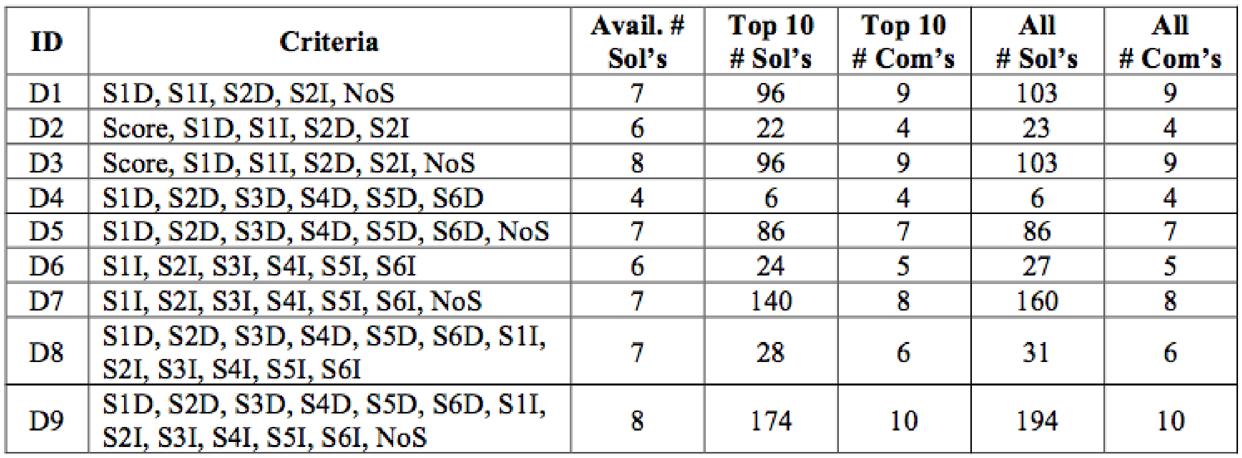
By comparing the different scenarios, we observe some general patterns. First, the top place finisher dominates across many scenarios. This is not surprising given this competitor finished with the highest overall score and a big margin of victory over the second place finisher, and when seeking general solutions the leaderboard scoring for data competitions typically favors solutions that offer a good balance between multiple objectives of interest. Second, the richness of the PFs (i.e. the number of solutions on the PFs) generally increases as (a) the number of available solutions increases (as we move from Case 1 (with only the final submissions of the prizewinners) through Case 2 (all submissions from prizewinners) to Case 3 (all submissions from all competitors) for each scenario), and (b) the number of criteria increases (as we move from group A to D of various scenarios). Third, the number of competitors associated with the solutions on the PFs also grows as the number of criteria increases (as evidenced by more colors in the barplots as we move from two to 5+ criteria scenarios).

In addition, some general patterns about the strengths and weaknesses of each top competitor emerge from examining the barplots. For example, the top place finisher has dominating performance on detecting and identifying sources 2, 3, and 6 as well as controlling FPRs, but is not quite as competitive on detecting or identifying sources 1, 4, and 5. The 2nd place finisher shows strength in detecting sources 1, 3, and 4. The 4th place finisher does well identifying sources 4 and 5. The 5th and 7th place teams are competitive in detecting source 5 and identifying source 1. These observations can help subject matter experts understand the strength and weaknesses of each competitor's solutions and choose the best algorithm for specific scenarios where only a subset of the criteria are of primary interest.



| ID | Criteria | Avail. # Sol's | Available Ranges | Top 10 # Sol's | Top 10 # Com's | Top 10 Ranges | All # Sol's | All # Com's | All Ranges |
|---|---|---|---|---|---|---|---|---|---|
| C1 | Score | | [68.64,76.43] | | | [61.14,76.74] | | | [47.79,76.74] |
| | S1D | | [0.426,0.574] | | | [0.415,0.625] | | | [0.373,0.625] |
| | S1I | | [0.092,0.397] | | | [0.084,0.444] | | | [0.084,0.461] |
| | NoS | 6 | [0.976,0.999] | 69 | 7 | [0.952,1] | 75 | 9 | [0.951,1] |
| C2 | Score | | [69.67,76.43] | | | [71.21,76.74] | | | [71.21,76.74] |
| | S2D | | [0.395,0.594] | | | [0.447,0.621] | | | [0.447,0.621] |
| | S2I | | [0.310,0.477] | | | [0.377,0.494] | | | [0.377,0.494] |
| | NoS | 5 | [0.983,0.999] | 25 | 2 | [0.951,1] | 25 | 2 | [0.951,1] |
| C3 | Score | | [69.67,76.43] | | | [69.77,76.74] | | | [69.77,76.74] |
| | S3D | | [0.455,0.629] | | | [0.454,0.682] | | | [0.454,0.682] |
| | S3I | | [0.189,0.324] | | | [0.204,0.348] | | | [0.204,0.348] |
| | NoS | 5 | [0.983,0.999] | 27 | 3 | [0.951,1] | 27 | 3 | [0.951,1] |
| C4 | Score | | [68.64,76.43] | | | [66.67,76.74] | | | [66.67,76.74] |
| | S4D | | [0.287,0.470] | | | [0.282,0.506] | | | [0.282,0.506] |
| | S4I | | [0.122,0.287] | | | [0.099,0.300] | | | [0.099,0.300] |
| | NoS | 7 | [0.976,0.999] | 56 | 6 | [0.952,1] | 56 | 6 | [0.952,1] |
| C5 | Score | | [69.67,76.43] | | | [62.59,76.74] | | | [61.34,76.74] |
| | S5D | | [0.548,0.747] | | | [0.532,0.777] | | | [0.531,0.777] |
| | S5I | | [0.371,0.567] | | | [0.445,0.629] | | | [0.445,0.629] |
| | NoS | 5 | [0.983,0.999] | 46 | 5 | [0.951,1] | 49 | 6 | [0.951,1] |
| C6 | Score | | [68.64,76.43] | | | [71.53,76.74] | | | [71.53,76.74] |
| | S5D | | [0.825,0.915] | | | [0.820,0.933] | | | [0.820,0.933] |
| | S5I | | [0.221,0.775] | | | [0.693,0.779] | | | [0.693,0.779] |
| | NoS | 4 | [0.983,0.999] | 25 | 1 | [0.951,1] | 25 | 1 | [0.951,1] |
| C7 | S1D | | [0.511,0.574] | | | [0.560,0.625] | | | [0.560,0.625] |
| | S1I | | [0.092,0.397] | | | [0.292,0.444] | | | [0.292,0.461] |
| | S2D | | [0.393,0.594] | | | [0.568,0.621] | | | [0.561,0.621] |
| | S2I | 5 | [0.261,0.477] | 6 | 2 | [0.452,0.494] | 7 | 3 | [0.452,0.494] |

Figure 70: Results for 4-criterion Pareto fronts, based on different collections of the solutions.



| ID | Criteria | Avail. # Sol's | Top 10 # Sol's | Top 10 # Com's | All # Sol's | All # Com's |
|---|---|---|---|---|---|---|
| D1 | S1D, S1I, S2D, S2I, NoS | 7 | 96 | 9 | 103 | 9 |
| D2 | Score, S1D, S1I, S2D, S2I | 6 | 22 | 4 | 23 | 4 |
| D3 | Score, S1D, S1I, S2D, S2I, NoS | 8 | 96 | 9 | 103 | 9 |
| D4 | S1D, S2D, S3D, S4D, S5D, S6D | 4 | 6 | 4 | 6 | 4 |
| D5 | S1D, S2D, S3D, S4D, S5D, S6D, NoS | 7 | 86 | 7 | 86 | 7 |
| D6 | S1I, S2I, S3I, S4I, S5I, S6I | 6 | 24 | 5 | 27 | 5 |
| D7 | S1I, S2I, S3I, S4I, S5I, S6I, NoS | 7 | 140 | 8 | 160 | 8 |
| D8 | S1D, S2D, S3D, S4D, S5D, S6D, S1I, S2I, S3I, S4I, S5I, S6I | 7 | 28 | 6 | 31 | 6 |
| D9 | S1D, S2D, S3D, S4D, S5D, S6D, S1I, S2I, S3I, S4I, S5I, S6I, NoS | 8 | 174 | 10 | 194 | 10 |

Figure 71: Results for 5 or more criteria Pareto fronts, based on different collections of the solutions.

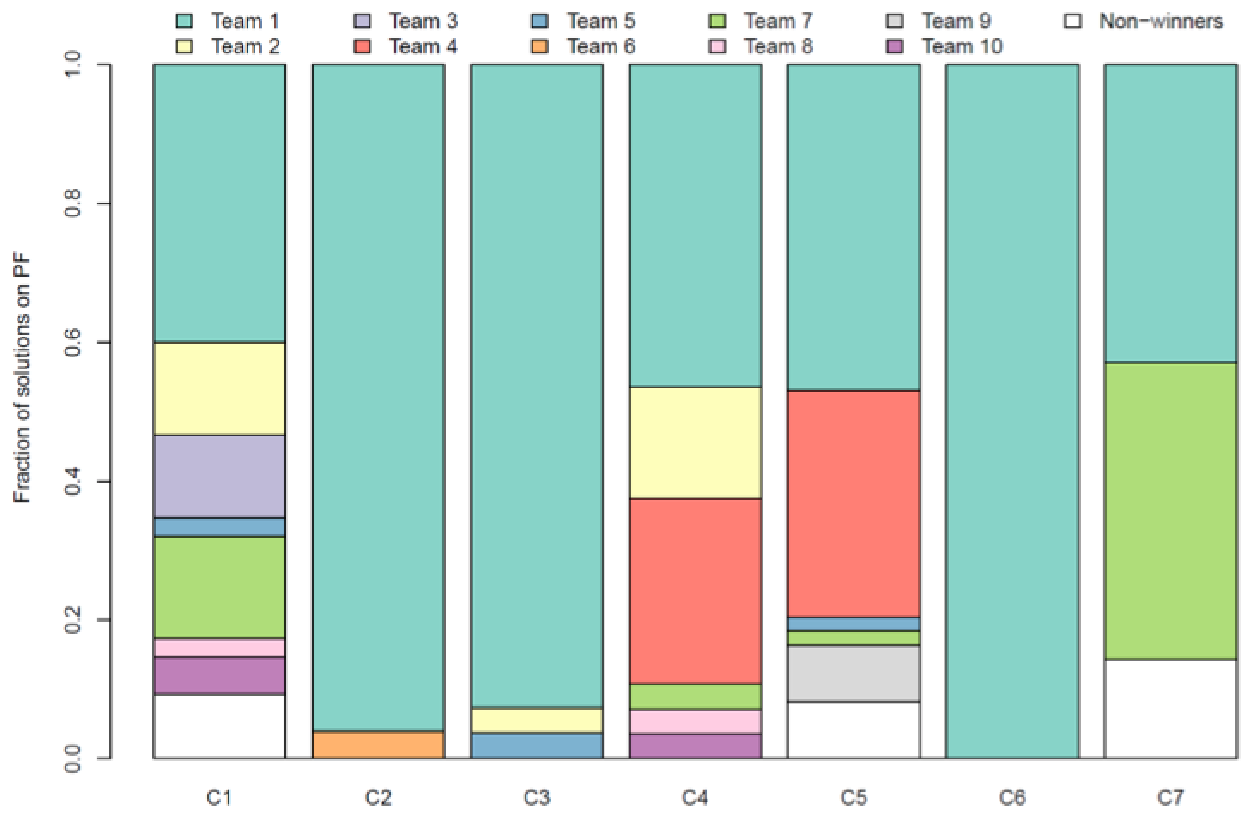


Figure 72: Barplot of the representation of different competitors on the Pareto front for each of the 4-criterion scenarios.

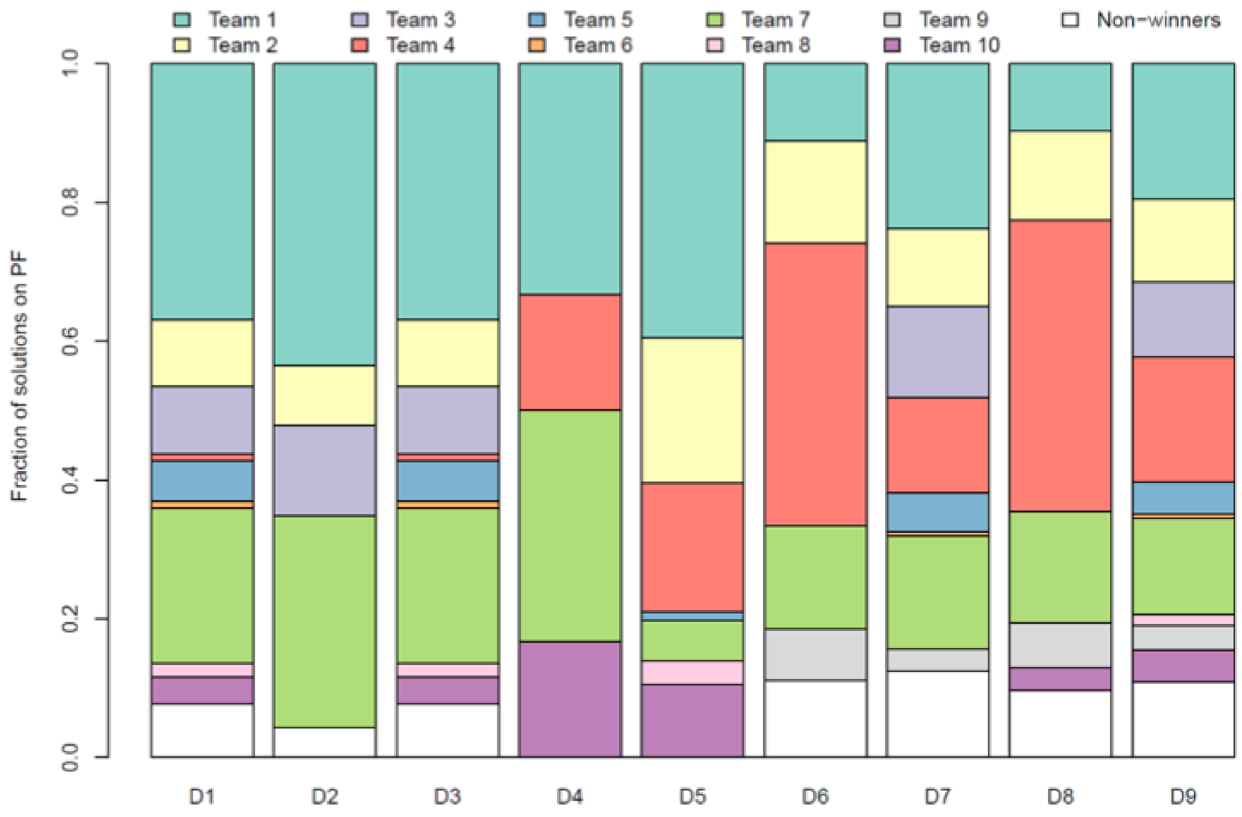


Figure 73: Barplot of the representation of different competitors on the Pareto front for each of the 5 or more criteria scenarios.

# 7 Description and Exploration of the Winning Algorithms

In this chapter we explore the approaches used by the top TopCoder competitors. The top 10 competitors were eligible for cash prizes with the requirement that they provide their working algorithms within a Dockerized image along with a writeup describing their methodology. We have then used this material to evaluate different approaches that yielded good results, identify best practices, and to consider whether portions of these algorithms merit further study.

It should be noted that the quality of the code within the Dockerized images, as well as the quality and clarity of material in the writeups, varied widely. Some of the competitors followed best practices of commenting and organizing their code, while also using clear input configuration files to modify their algorithms, enabling deeper study and perturbative analysis. In this chapter we first provide an overview of the scores and methods used by the competitors, we then provide a detailed examination of the top 5 prize-collecting competitors approaches. We then discuss a high-level evaluation of the top algorithm and best-practices that were followed. Finally, we describe several model-introspection analyses that were performed to explore and understand what the Machine Learning methods employed were learning, how perturbation of algorithm configurations impacts scores, and thoughts regarding "deploy-ability" or "field-ability" of these algorithms or modifications of them.

## 7.1 High-level summary of the Top Ten

The top ten submitted algorithms were selected for analysis and comparison. Two of the top scoring algorithms were not submitted for prizes so lowering scoring competitors were promoted into the top scoring leaderboard as shown in Table 7. Within these leaderboard, several groups emerged as ranked by their final score (and by the bootstrapping analysis presented in the previous chapter). These four groups are delineated by the dashed lines in Table 7. The competitor submitted writeups are included in the appendices for reference.

Table 7: The final top 10 competitor leaderboard. Two competitors (marked with "*") placed in the top 10 but did not submit their algorithms so the lower scoring competitors made the top 10. The final score shows 4 clusters of competitor's performance which we note with with dashed lines. Source: `https://www.topcoder.com/challenges/30085346?tab=submissions`.

| Final Rank | Provisional Rank | Username | Writeup Appendix | Final Score | Provisional Score | Decision Tree(s) | Neural Net(s) | Likelihood Testing |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | `pfr` | A | 76.4289 | 90.26718 | | ✓ | |
| 2 | 7 | `p_kuzmin` | B | 73.6693 | 86.67468 | ✓ | ✓ | |
| 3 | 3 | `gardn999` | C | 73.42755 | 87.45759 | ✓ | | |
| 4 | 2 | `rayvanve` | D | 71.83693 | 87.88757 | ✓ | ✓ | |
| 5* | 15 | `pkbrk9` | — | 71.82177 | 82.47646 | — | — | — |
| 6 | 6 | `cyril.v` | E | 71.7367 | 86.82956 | | ✓ | |
| 7* | 5 | `mike.grosskopf` | — | 71.47615 | 87.12627 | — | — | — |
| 8 | 4 | `wleite` | F | 71.38198 | 87.41077 | ✓ | | |
| 9 | 9 | `smg478` | G | 71.33487 | 85.62803 | ✓ | | |
| 10 | 11 | `cannab` | H | 69.66863 | 84.57155 | | ✓ | |
| 11 | 8 | `pasda` | I | 69.42755 | 85.69036 | | | ✓ |
| 12 | 21 | `ZFTurbo` | J | 68.63695 | 78.64455 | | ✓ | |

The top competitors implemented a wide–range of cutting–edge machine learning approaches using open source frameworks to architect their algorithms. The majority of these algorithms use recent advancements in artificial neural networks which leverage hardware acceleration on GPU's to train and evaluate results. In addition, some competitors used decision trees to process the output of the neural network(s) while others used them for a stand–alone solution. Almost every competitor used some ensemble of independently trained neural nets and/or decision trees with a final solution calculated from combined outputs as shown in the high level overview in Figure 74. It is worth noting that while 9 out of 10 of the submitted algorithms utilized general machine learning approaches, `pasda` (in 11th place) used a statistical likelihood estimate from the listmode data to detect, identify and localize. A summary of this approach can be found in Table 8.

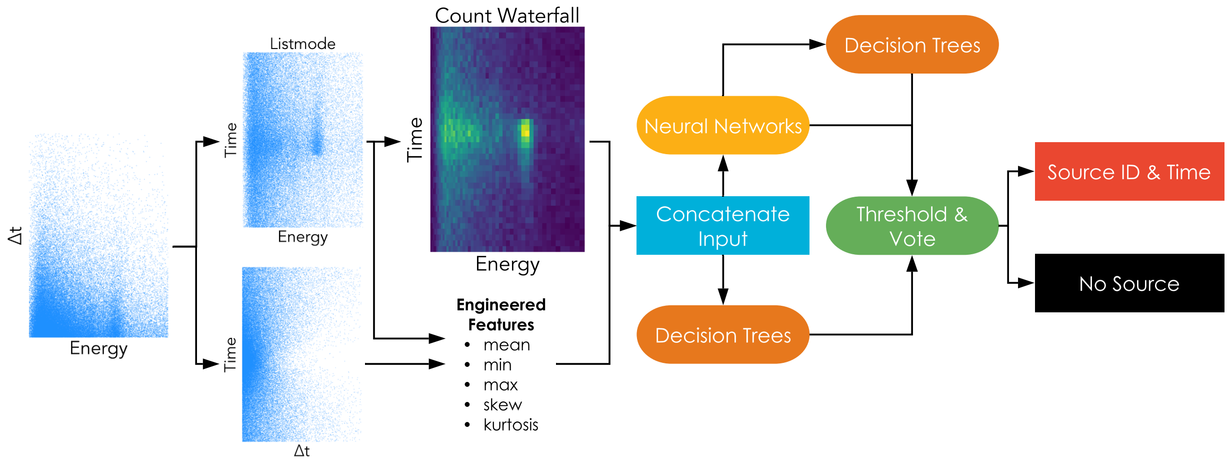


Figure 74: General approach used by 9 out of 10 top competitors.

Many neural network architectures were implemented which ranged from custom convolutional networks like that of `pfr` (in 1st place) to more general networks developed for object detection in images: UNET [101], ResNet50 [75] and DenseNet121 [102]. Three competitors attempted to use neural networks with some memory to leverage the time evolution of each spectra during the run but all concluded that their convolutional approaches performed better and required less hyper–parameter optimization when this memory was removed. Of the convolutional networks implemented, the best performing were single dimension kernels in energy which were convolved along the time dimension of the input waterfalls. This is also true for the highest performing 2D convolutional layers which has kernel sizes that almost spanned the entire energy bin range. All neural network approaches use one of the more common frameworks with a Python interface: PyTorch or Keras (with a Tensorflow back–end). Most competitors implemented techniques to avoid network over–fitting while training. These included changes to the network architecture like dropout to reduce the number of features between convolutional layers, input regularization (L1 normalization and mean–subtraction) and output regularization (softmax and log–softmax).

The decision trees used by competitors fall into two categories (1) random forests [103] and (2) gradient boosted machines [104]. Both are typically used for categorization tasks (like source detection and identification) and don't require a discrete GPU like neural networks. Two competitors (`gardn999` and `wleite` ) used pure Java implementations of binary tree random forests for their entire approach. Both built a forest for each source type which processed input spectra (and engineered features in the case of `gardn999` ) and returned weights per each source over each integration time. Other competitors (`p_kuzmin` , `rayvanve` , and `smg478` ) used Light Gradient Boosted Machines (LGBM's) [104] in conjuction with neural networks to process input data and/or the neural network output for further classification. LGBM's are a relatively new decision tree implementation which is heavily optimized for parallel CPU and GPU computation and offers a convenient Python package for simple passing of data between neural networks and the boosted machines.

While the design of each algorithm differed, the competitors selected similar data representations for their algorithm input(s). With the exception of `pasda` (as mentioned above) all competitors chose to discretize the listmode event data into energy spectra for constant integration times through each run. Each competitor chose a different energy bin structure with most opting for bin widths that increased with energy to mimic the increasing energy resolution (and decreasing count-rate). This is a commonality with classical template–based detection and identification algorithms which also use bin structures which aim to make the ratio of energy bins per energy resolution constant across the spectrum. These spectra were analyzed individually or more commonly as a "waterfall" 2D histogram to leverage the energy and time information in the listmode data simultaneously. The waterfall matrix allows the application of approaches from other fields since this data representation is analogous to a grayscale image or an acoustic wave spectrogram. Since the input to neural networks and decision trees is typically a fixed size (number of time and energy bins) competitors either fixed the integration time and analyzed a time window of data across each run or fixed the number of time bins such that each run was always the same input size and the integration times varied. Before providing the spectra to neural networks or decision trees, several competitors applied smoothing kernel (*boxcar*, *gaussian*, or *ricker*) to reduce noise between adjacent bins at the expense of the intrinsic counting statistics. Only `pfr` use binomial resampling of each bin to scale the spectra while maintaining the underlying count variation. In addition to the energy spectra vs time input, multiple competitors engineered statistical features from the data which were provided alongside the histogrammed events.

As previously discussed, the algorithm objectives were to determine the presence of a source in each run and, if so, identify the source type and time of closest approach. Since the competition scoring gave partial credit to source detection and incorrect identification, the common approach was a multi–tiered algorithm which first detected the presence of any source before determining the source ID or time. Some built completely separate components to solve each task such that the first objective as binary classification, the second was 6 source classification and the third was timing determined from source likelihoods over time. Instead of this more complex approach, most built a single algorithm which would output 6 source likelihoods (weights) for each time window and aggregate these for detection and identification tasks. This allowed detection even in cases where the energy spectrum was significantly perturbed due to environmental scattering such that identification was difficult. While the simple binary or six label classification tasks have clear targets for training (one for the correct class and zero otherwise), the training targets for the time–dependant source weights proved more challenging. While all the time bins could be given the same weight for a training run with a source, each competitor took a more prudent approach of training against a target vector which maximized the source weights at the time of closest approach. Some took this a step further to determine the width (number of time bins around the time of closest approach) in which they declared the source "present".

Another commonality between the top approaches was the prioritization of detection and identification which multiple competitors referred to as the easier objective. The time of closest approach determination seemed to be an after–thought for most competitors who determined it was likely easier to optimize for accurate detection and identification instead of timing. Nearly all competitors used the maximum or a weighted average of the time dependant source weights to determine the closest approach time. The accuracy of this method was influenced by the time dependant training labels mentioned above and, to a larger extent, the count rate asymmetry of some of the runs due to large occlusions before or after the time of closet approach. These asymmetries caused the weighted average of time by source likelihoods to skew away from the true time of closest approach.

Most competitors used an ensemble of networks and/or decision trees (up to 30 as with `pfr` ) which were trained on different portions of the training data to harden the approaches against over–fitting. This was a clear advantage in the higher performing competitors while several of the lower performers mentioned they likely would implement this in future work. A couple competitors also reversed the driving direction to further improve the diversity of the training data when repeating training with the same (or overlapping portions of) training data.

For inference (executing the algorithms) the testing data were pre–processed and smoothing (if used for training) was applied. While the training labels may have been ones and zeros, the output from most of the aforementioned approaches was a normalized range. In a classical algorithm, a threshold for detection and identification would be determined from processing many background (source free) runs and choosing a threshold based on a desired false alarm rate (FAR). None of the competitors took this approach (although some mentioned it in their write–up) and instead set the thresholds empirically based on trial and error with some limited probing of the leaderboard to optimize for their provisional score.

A detailed summary of the top 10 approaches is provided in Table 8.

Table 8: Winning Algorithms Overview.

| **Competitor** | **Hardware/ Software Requirements** | **Uses Source Templates** | **Uses Speed/ Offset** | **Train/ Test Hours** | **Input Data Processing** | **ML Approach** | **Cross Validation** |
|---|---|---|---|---|---|---|---|
| `pfr` | Nvidia GPU<br>Python<br>PyTorch | | ✓ | 7.1/0.2 | 40, 80, 160 time bins<br>Non–linear energy bins | Ensemble of 1D CNN's (x30)<br>3 time scales<br>Thresholded output | Ten 90% splits repeated 3 times<br>Driving direction reversed<br>Binomial resampling of counts |
| `p_kuzmin` | GPU<br>Python<br>Keras | | | 1.8/0.4 | 1 second 50 keV binned spectra<br>Listmode statistical features | 3 1D CNN or MLP Ensembles<br>LGBM classifiers | 5 training splits |
| `gardn999` | CPU<br>Java | | | 0.6/0.3 | Square root energy bins<br>0.5s time bins<br>Listmode statistical features | Random forest per source | 8 training splits |
| `rayvanve` | CPU or GPU<br>Python<br>Keras | ✓ | | 12.5/11.8 | Multiple energy bin structures<br>Multiple constant time bin widths<br>2D gaussian smoothing | Compare spectrum to source template<br>2D CNN over time window<br>Random forest classifiers | No training splits<br>Feature selection for RF's |
| `cyril.v` | GPU<br>Python<br>Keras | | | 3.8/0.4 | Constant 0.5 s time bins<br>Log(E) bin widths | 1D CNN's passed into LSTM<br>Model 1 for det/ID<br>Model 2 for coarse time<br>Model 3 for fine time | Driving direction reversed<br>10 CV splits during training |
| `wleite` | CPU<br>Java | | ✓ | 11.8/0.9 | 800 energy bins<br>3 time binnings (overlapping)<br>Ratio of adjacent time bins<br>Boxcar smoothing | Random Forest per source/time<br>Det/ID from top time score<br>Timing from weighted average | 50% training splits<br>Separated 1500 "harder" cases |
| `smg478` | GPU<br>Python<br>Keras | ✓ | | 1.8/7.1 | 30keV energy bins<br>3 time binnings<br>(fixed events, 1/30 total time)<br>Peak–to–peak ratios<br>Peak–to–compton ratios | 5 CNN+MLP Models<br>7 Output Classes<br>Det/ID from voting<br>Coarse timing from max NN output<br>Fine timing from max peak counts<br>Tried LGBM and LSTM (not used) | 5 training splits |
| `cannab` | Nvidia GPU<br>Python<br>PyTorch | | ✓ | 84.0/4.3 | 81 Time Bins<br>81 Energy Bins<br>Listmode features | 1D DPN68 [105] for Det+ID and speed<br>1D U-Net [101] for time<br>Averages ensemble and applies threshold<br>Mentions LGBM (not enough time) | 2 training splits<br>Shift/scale listmode features |
| `pasda` | CPU<br>R | ✓ | | 0.2/2.9 | Source templates to 0.5keV PMF's<br>50/50 mix of source 6<br>Background 0.5 keV spectrum PMF<br>Source training data not used | Source–to–background probability ratios<br>Ratios for each event and source<br>Kernel regression (across time)<br>Source/Time from max statistic<br>Set threshold by leaderboard probing | n/a |
| `ZFTurbo` | GPU<br>Python<br>Keras | | | 51.6/22.1 | Raw energy and time used<br>Listmode features<br>32768 events per input | 3 Conv1D models for Det/ID<br>RNN model for timing<br>UNET/ResNet50/DenseNet121<br>Outputs averaged and thresholded | Used to set threshold |

## 7.2 Deep dive into the top five algorithms

In the previous section we provided an overview of the top ten competitors' approaches. Here, we aim to gain a better understanding of the top 5 approaches through the dissection of each algorithm, starting with the submitted report and then reading the source code. We discuss the details of the input data preparation, the machine–learning techniques used, the algorithm structure, the training methodology and the output post-processing to finalize results. We compare these approaches to best practices from subject matter experts 2.2. We identify machine–learning best practices which were used and point out those that were not. Finally, we discuss each algorithm's potential (or lack thereof) for modification to meet operational deployment requirements.

Our workflow in performing this assessment included using multiple software development and analysis tools. TopCoder provided each competitor's code, report and Dockerfile in a GitLab repository which were each forked into a development group (`https://gitlab.com/urban-radiation-detection-dev`) for further analysis. Each repository's development branch was used to add tools like Jupyter notebooks and neural network introspection libraries to facilitate analysis of each stage of the algorithm. We used Docker to quickly reproduce the development environment for each competitor to rerun their algorithm and analyze the components in place. These portable environments (containers) could then be used for local development before before deploying on Amazon Web Service (AWS) machines enabled with high performance GPU's necessary to train the neural networks.

This approach proved successful for some algorithms (most notably the first place approach from `pfr` ) whose source code could be read while following the write–up as a guide. For a subset of the competitors our introspection was limited by an inadequate write–up, sloppy code and/or ad–hoc combinations of methods and concepts. The results of our analysis are presented in the following sections.

### 7.2.1 01-pfr

The first place competitor (`pfr` ) submitted a well organized and flexible algorithm to rise to the top of the leaderboard by a healthy margin. In fact, `pfr` only joined the competition in the final 36 hours and their first submission would have still beaten all other competitors. They implemented a large (30 network) ensemble of multi–layer convolutional neural networks with robust data preparation and augmentation to avoid over–fitting to the labelled training data and ensure a more generalized final product. They used PyTorch to construct their networks and Numpy/Pandas to prepare the input and post–process the output into a final solution. The submission is divided into multiple components: (1) raw CSV listmode pre–processing into binary formatted listmode data rounded to 2keV, (2) a loop to train 30 individual neural networks across a fixed energy binning and 3 time binnings, (3) a loop to execute the ensemble of networks with the testing data, and (4) a final aggregation step to process the network ensemble output for detection, identification, and timing. All of these components are structured into a local Python library with a set of structured tools for executing each stage with local code reuse to avoid unnecessary repetition. Finally, the algorithm hyper–parameters (network size, number of convolution layer features, training configuration, etc) was controlled from a straightforward YAML specification.

Like most other competitors, `pfr` converted the raw listmode event data to a energy vs time "waterfall" histogram. Since the first 30 seconds of data were known to not contain a source, this data was dropped from all further analysis. They used a non–linear energy bin structure of 186 bins from 0–2.811MeV with an overflow bin. Instead of using a computationally expensive histogram operation, they grouped the energies starting with the base binning of 2keV from the pre-processing stage. The energy binning was then defined as regions of increasing combinations of the base binning (2keV, 4, 8, etc) as shown in Figure 75 They took a unique approach to assuring the same input shape for the network by fixing the number of time bins instead of the integration time. This meant that the input waterfalls has different integration times based on the duration of each run but the same input size so they could be batch processed together.

The waterfalls were stacked into batches (as is typical for efficient training and executing neural networks) and were input to a neural network with a sequence of 1D convolutional layers with activation between layers. The convolution layers consisted of multiple feature kernels of size 3 (time bins) by 186 energy bins by number of features. The first (so–called embedding) layer converted the count data across energy bins and 3 time bins to 32 features (an example of the layer convolution kernel is shown in Figure 82.) These 32 features were then feed into a sequence of 6 1D convolution layers with 40 features followed by rectified linear units

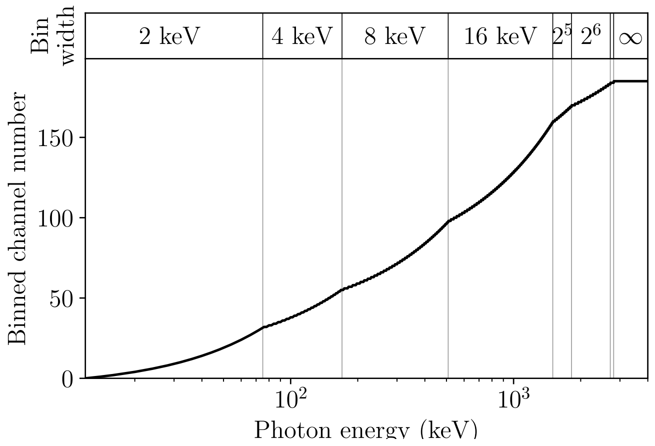


Figure 75: Non–linear energy bin structure used by `pfr` .

(ReLU) as the source of non-linearity. The final stage was another 1D convolution layer with 6 features to represent the likelihood of each source at each time bin.

This network, shown in Fig. 76, was used for 3-time binnings starting from the initial 160 bins, then 80 and 40. The usage of multiple time structures allowed the same neural network to be sensitive to encounters at different speed to standoff ratios in which the counts from the source encounter varied in duration. The output from network for each input were flattened such that the length of each network output was the number of time bins times 6 sources. These outputs were concatenated with a one appended at the end before application of a LogSoftMax normalization. The extra entry enforced small LogSoftMax values for the other 1680 entries when a source was not present.

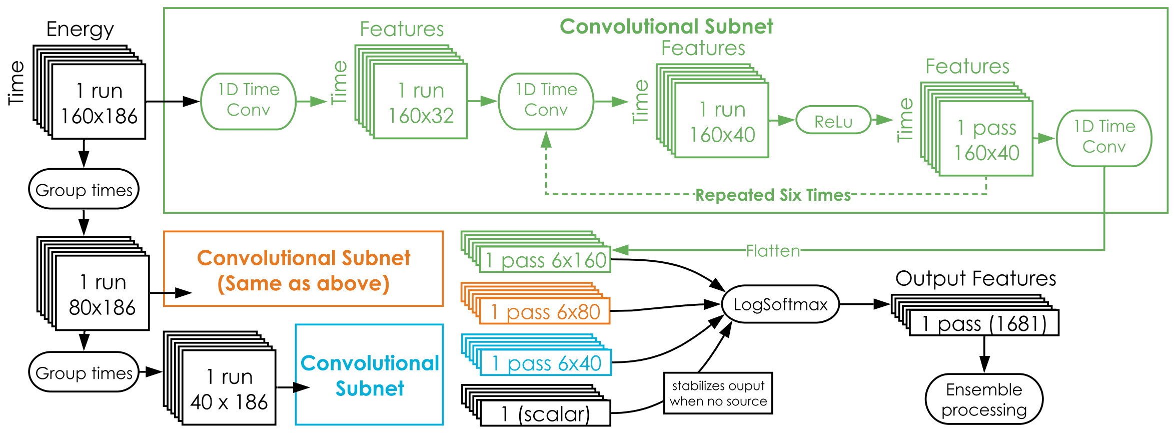


Figure 76: Schematic of the first place algorithm including the multi–layer 1D CNN neural network at different time scales and final classification based on the output.

For the source training cases, the target weights were set to a normalized boxcar centered at the time of closest approach for only one of the integration times and the appropriate source. The width of the boxcar and integration time were chosen by a heuristic which included the speed/offset ratio from the provided scoring data which only two other competitors took advantage of (`wleite` and `cannab` ). The target for background only cases contained all zeroes except for the final entry.

A total of 30 networks were different splits and augmentations applied to the labeled training data. The data was split into 10 groups of 90% training and 10% validation. These groups were not stratified (divided into even numbers of each class) which may have decreased performance since each network was trained on different ratios of each target class. Each group was use to train 3 different networks. Each waterfall had it's counts resampled from a binomial distribution with a success ratio chosen from 0.25, 0.6 or 1.0. Additionally, the direction of the waterfall was randomly reversed. Training was performed using Kullback–Leibler divergence to calculate loss for 40 epochs with a learning rate decreased after epoch 20 and 30. No early stopping was implemented which may have decreased performance since the final network may have had a larger loss metric than that of earlier epochs.

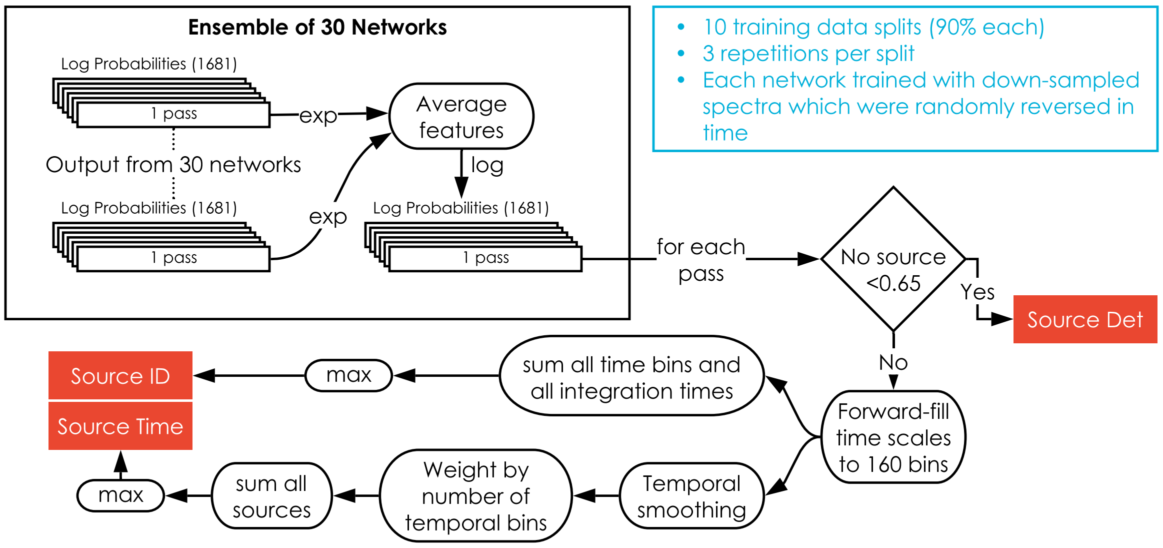


Figure 77: Ensemble of networks in the first place algorithm from `pfr` .

The final ensemble of 30 networks were used in sequence for inference on test data (Figure 77). Each network reported 1681 features from the final LogSoftMax (as shown in Fig. 76) which were exponentiated (into probability space), averaged across the ensemble and then transformed back into log–space. Since the following analysis occurred in probability space instead of log space, the author has omitted the explicit conversion from log to probability space by the exponential operation. The final entry (which was hard–coded to 1 before the LogSoftMax) was used to determine the presence of a source. If the final entry was greater than 0.65, no source detection was reported. Otherwise, the remaining 1680 source probabilities were expanded to the finest integration time (the largest number of time bins) via repetition (upsampling) before source ID and time determination. The expanded probabilities were then summed across all integration times and time bins to determine the most likely source ID. The competitor noted that the ID determination would have made more sense with the unexpanded probability but went on to state that the effect should be negligible. To find the time of closest approach, the expanded source probabilities for each integration time were weighted by the inverse of the square of the number of time bins vs coarsest number of time bins (i.e. the 160 time bins probabilities were scaled by 1, the 80 time bins were scaled by 1/4 and the 40 time bins were scaled by 1/16). The re-scaled probabilities improved the determination of closest approach time by emphasizing the finest integration times with the highest timing precision. The expanded and scaled probabilities were then summed across different times and source ID's, followed by smoothing with a Gaussian kernel with $\sigma = 7$ time bins. Finally, the center of the time bin with the maximum value was reported as the source time.

### 7.2.2 02-p-kuzmin

In contrast to the first place approach from `pfr` , `p_kuzmin` took a more ad–hoc approach with a more complex two stage algorithm with different ensembles for detection, identification and timing in the second stage. The stages used both neural networks implemented with keras/tensorflow and light gradient boosted machines (LGBM's). Unlike `pfr` 's approach, `p_kuzmin` also used features engineered from the listmode data.

Some pre–processing was applied to the raw listmode input to extract features to use for algorithm inputs. The data was aggregated to 1Hz with 49.575keV energy bins (FWHM at 661.7 keV) up to 3.2MeV (the competitors note that there are few events beyond this range). If the run was found to contain a source, the next level 2 component determined the time of closest approach.

Within each of these 1Hz segments, multiple statistical features were engineered from the listmode data including the mean, max, min, median, standard deviation, and skew of the (1) time between events and (2) energy between events, as well as the gross counts in 1 second window minus 1 (the delta times were counted.) A diagram of this approach is shown in Fig. 78 and described in more detail below.

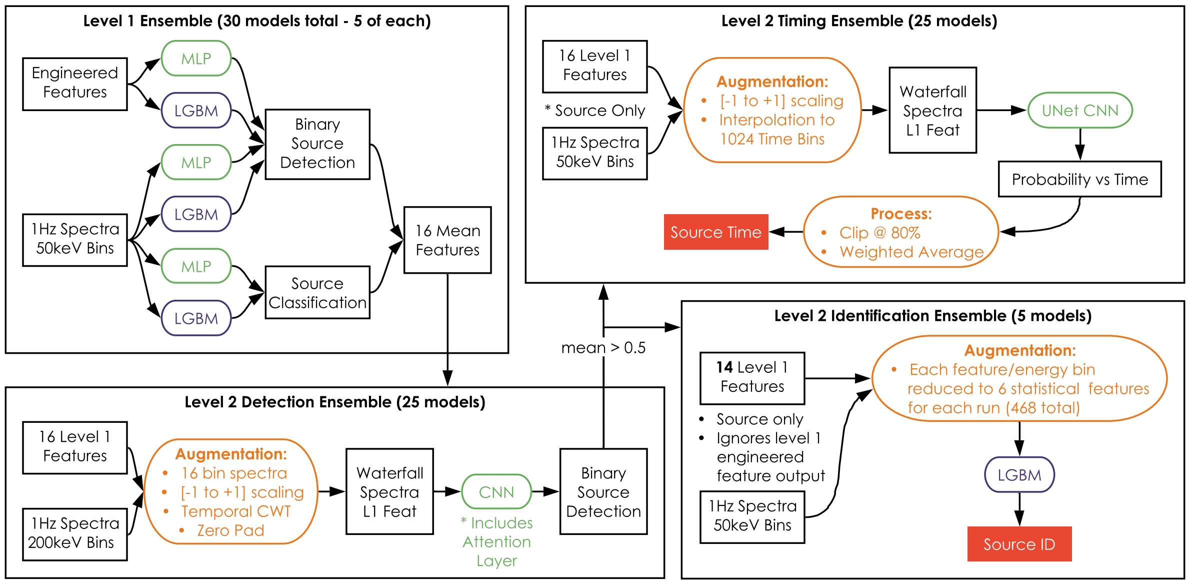


Figure 78: Schematic of the second place algorithm from `p_kuzmin` .

The first stage was an ensemble of multi–layer perceptrons (MLP's) and LGBM's. The 1Hz engineered features (13 total) and spectra (64 bins) were used with separate models for source detection and classification (ID). The engineered features were used in a four layer MLP with decreasing features (128, 64, 32, 6), ReLu activation between the layers and a final sigmoid activation on the final binary output. Additionally, the engineered features were used by a LGBM to also predict the presence of a source. The spectra were used in a MLP and LGBM to output a detection likelihood as well as another MLP and LGBM pair to classify the source ID (6 normalized outputs). The spectral detection MLP has the same structure as the one for engineered features. The spectral identification MLP also had the same structure except for the final layer which output 6 features with a SoftMax. Five of each model were trained against binary labels for a total of 30 level 1 models. The detection and identification outputs from each set of 5 models were averaged and the means were concatenated for a level 1 output vector of 16 features for every 1 second time bin across all runs (1292411 samples).

The first of three components for the second stage was designed to detect the presence of a source before continuing to the identification and timing stages. The spectra were reduced to 16 bins (200keV each) and concatenated with the 16 level 1 features. Each of the 32 features were re-scaled between [-1, +1] using the minimum and maximum across all 1292411 time bins. The re-scaled features for each run were then smoothed with the maximum of a temporal continuous wavelet transform with widths of 1–30 channels.

Lastly, the waterfall (time-series) of features for each run was zero padded to 1024 time bins (the final shape per run was $1024 \times 32$). This waterfall was passed to a multi–layer 1D convolutional neural network (over time) with activation, dropout and an attention layer [106]. The output of this network was a binary detection with sigmoid activation for each of the 1024 time bins per run. The run was determined to contain a source if the mean of the time dependant binary detections was greater than 0.5 (this was an empirical threshold based on trial and error).

If the run was found to contain a source, the next level 2 component determined the time of closest approach using the same level 1 features and spectra used in the previous component. The 16 features and spectra of 64 energy bins were concatenated and re-scaled between [-1, +1] using the minimum and maximum across all 1292411 time bins. Approximately half of the training runs were cropped to a random time at least 30 seconds before the source time and at least 10 seconds after. If the run was shorter than would allow for this time padding, the min/max time was used. Then the features and spectra for each run were interpolated (up–sampled) to 1024 time bins such that the effective integration time was less than 1 second. The resultant feature and spectra waterfall was input to a 1D UNet CNN [101] with internal features increasing from 64–1024 and then decreasing down to 1 feature for each time bin. All Conv1D kernels have a width of 3 time bins except for the final layer, the LeakyReLu function is used as activation between Conv1D layers, 50% dropout was used between some layers and the final output was activated with a sigmoid function and values below 80% of the maximum were set to zero. The predicted time was calculated from an average of the time bin centers weighted by the network output. For training, the output target was generated from a normal PDF, centered on the ground–truth time, which was masked at a randomly selected threshold between 0.004–0.24. This resulted in a boxcar target of ones around the ground–truth with a half–width of 1.7–3.3 seconds and zeros elsewhere. Training was performed with stopping criterion determined by average of a per–run score calculated from the cross–validation output. This score was -2 for times farther than 4 seconds from the ground–truth or calculated from a cosine function which peaked at the ground–truth and decreased to zero at $\pm 4$ seconds. The training learning rate was decreased by 40% every 2 epochs to a minimum of 1% the original rate. This training loop was stopped after 10 epochs without score improvement. Five models were trained for each stratified (equal label distribution) and shuffled 80/20 cross validation split for a total of 25 models. For testing, the predicted time was calculated from the average of the times weighted by the network output and any times below 26 seconds were set to 0.

The last of the level 2 components determined the source ID using the level 1 features and spectra like in the previous component. The level 1 features calculated from the engineered features were not included such that only 14 features were used. Metrics (mean, median, max, min, standard deviation, and skew) were calculated from the time–dependent 14 features and 64 energy bins. These 468 ($6 \times (14 + 64)$) features were fed into a LGBM with 6 outputs corresponding to each source type. One model was trained for each stratified (equal label distribution) and shuffled 80/20 cross validation split for a total of 5 models. For testing, the predicted source class was calculated from the maximum of the averaged outputs from the 5 models.

This approach was complex, contained many nuances not included or explained in `p_kuzmin` 's write–up and still resulted in a lower score than `pfr` . In contrast to `pfr` , `p_kuzmin` 's source code was heavily copied between scripts with minor modifications and contained numerous unused functions and misleading names (like LSTM for the 1D UNet). Previous approaches were scattered within the scripts as commented lines which gives some insight into failed approaches such as activation functions, dropout layers and max–pooling aggregation.

The final score for `p_kuzmin` jumped them to 2nd place from their 7th placement with the public score. This implies that `p_kuzmin` 's approach more accurately handled the more difficult source encounters than the other competitors in the second scoring group. This may be due to the independent level 2 components, however `pfr` had a significantly higher score and didn't take this approach. Since `p_kuzmin` only used one time binning for each run, their approach likely had difficulty generalizing for all detector speeds and source standoffs.

The competitor made some notable recommendations for future work including: (1) refactoring the code for more efficient data processing, (2) generating additional features from the Fourier transform of the listmode timestamps, (3) using the full CWT matrix per spectrum as input to a 2D CNN, (4) adding early stopping to the level 1 MLP training, and (5) searching for better aggregation windows than 1 second. The author also notes that future work would include upgrading to TensorFlow 2 which, like refactoring, would be a time intensive effort to remove the deprecated functionality and upgrade to the new API since Keras is now built into TensorFlow.

While this approach was one of the more difficult to analyze, it does offer some useful best practices:

1. Training neural networks with early stopping is superior to a fixed number of epochs and is essential for cross–validation.
2. Stratified and shuffled cross–validation data selection should be used to randomize the training data and assure each model is trained with a similar number of data from each class.
3. General neural network architectures can be applied to the radiation detection and localization domain. This is evident from the usage of a 1D UNet which was designed for feature segmentation in biomedical image analysis. In this case, the UNet is essentially tasked with segmenting the spectra (plus features) in the waterfall which contains the source.

#### 7.2.3 03-gardn999

The competitor `gardn999` used random forest classifiers with very good results, a schematic of this approach is shown in Figure 79. Input features to the random forests were generated primarily by histogramming listmode data. The bin edges were defined using 60 energy bins spaced following $\sqrt{2E}$ and half second time windows. Adjacent time-bins were then smoothed by including 1/4 of the counts from the previous and following bins. Additionally, two more features were appended to the histograms, the mean and maximum energy in each time-bin.

Six random forests were trained, one for each source type. Each forest consisted of 100 trees which each use 50% of the training data and 20% of the of features. Each tree was built by iterating over the training feature vectors starting at the base node of the tree. The node used 10 feature vectors to select the feature column and split value by iteration, after which two more nodes were created until the max number of nodes (20% of the total number of training data) was reached or the nodes ran out of run data. Positive time segments were labeled as those within 10 seconds of the time of closest approach and weighted with a parabolic function. All other time segments were labeled as negatives. Theses labels, along with the input features, were then used to train the random forests.

When making predictions, the random forests were used to analyze the data from a particular run. Once processed, the outputs were aggregated for detection/identification and timing. The detection/identification was performed by computing the maximum sum of any 7 adjacent time-bins for each source type. If the largest of these values was above a specified threshold then the source-type corresponding to that random forest was alarmed. If the threshold was not exceeded then no alarm was sounded. The time of closest approach was identified by selecting the 80 adjacent time-bins with the largest probability raised to the 7th power, and then computing a weighted average of the times, using probability raised to the 7th power as the weighting.

#### 7.2.4 04-rayvanve

The competitor `rayvanve` applied several different concepts in their algorithm including feature engineering, different distance measures using the supplied templates, convolutional neural networks, and random forests (Fig. 80). The data were pre-processed in several ways. The raw list–mode data was histogrammed with small energy and time bins followed by a 2D Gaussian smoothing of this "image" (or waterfall) with different energy and time kernel widths. A range of bin sizes were selected to cover broad range of source encounter times. The smoothed result was then combined into larger time and energy bins for further processing. The competitor generated 11 images (see Table 9 for details) to engineer features by computing distance metrics (Hellinger, cosine, L2) for each 1D energy spectrum (image row) from the competition provided source templates (except the 6th and mixed source which didn't have a template). These time dependant distances were then used to generate features by taking the max, min, mean, std, kurtosis, argmax, argmin, and snr both before and after mean subtraction.

Additionally, the competitor generated 2D histograms using two different energy binning schemes and constant time bins (see Table 10 for details). A sliding time-window was used to allow a fixed size 2D histogram to be used as input to the CNN. The first convolutional kernel was either the full length of the

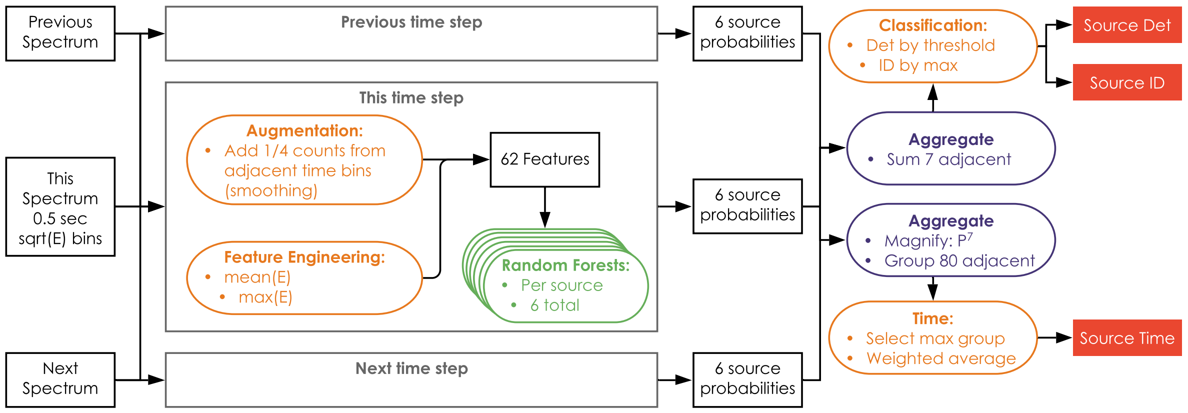


Figure 79: Schematic of the third place algorithm submitted by `gardn999` .

Table 9: Waterfall specifications for heuristic calculations in `rayvanve` 's approach.

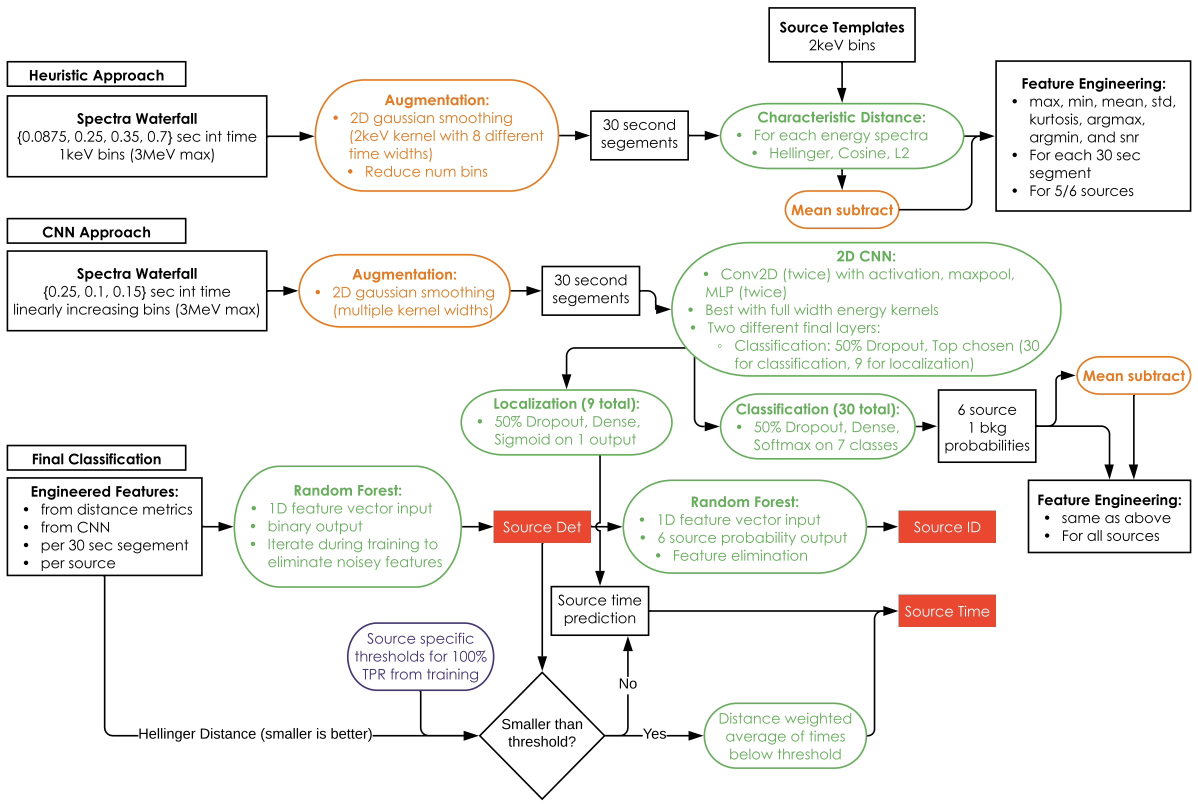
| **Competitor assigned ID** | **fine energy bin (keV)** | **energy bin (keV)** | **num energy bins** | **energy smoothing width (keV)** | **fine time bin (s)** | **time bin (s)** | **time smoothing width (s)** |
|---|---|---|---|---|---|---|---|
| 324484 | 0.50 | 1.0 | 3000 | 2.0 | 0.0875 | 0.0875 | 1.00 |
| 16799578 | 0.50 | 2.0 | 1500 | 2.0 | 0.0625 | 0.2500 | 1.25 |
| 19279309 | 0.50 | 2.0 | 1500 | 2.0 | 0.0875 | 0.3500 | 1.50 |
| 21602281 | 0.50 | 2.0 | 1500 | 2.0 | 0.0625 | 0.2500 | 0.75 |
| 22517282 | 0.75 | 3.0 | 1000 | 2.0 | 0.0875 | 0.3500 | 1.00 |
| 30072664 | 0.50 | 2.0 | 1500 | 2.5 | 0.1750 | 0.7000 | 3.00 |
| 31335721 | 0.50 | 2.0 | 1500 | 2.0 | 0.0625 | 0.2500 | 0.25 |
| 33815452 | 0.50 | 2.0 | 1500 | 2.0 | 0.0875 | 0.3500 | 0.50 |
| 40634754 | 0.50 | 1.0 | 3000 | 2.0 | 0.0875 | 0.0875 | 0.25 |
| 45383928 | 0.50 | 1.0 | 3000 | 2.0 | 0.0875 | 0.0875 | 0.15 |
| 45501474 | 0.50 | 1.0 | 3000 | 2.0 | 0.0875 | 0.0875 | 0.50 |

spectrum or 3 bins less, similar to that of the first place approach. The 2D CNNs had two convolutional layers with 64 features followed by batch normalization after the first and max pooling after both. The outputs were then flattened before 50% dropout followed by a dense layer with softmax activation and 7 outputs including background. Training was performed for up to 6 waterfalls per training run depending on the run length and the network input size. Thirty networks were trained to classify a source in the waterfall with the training labels for the correct class set to 0.2 for no source, 0.07 for a run with a source that wasn't in the waterfall and 1 for a source in the waterfall. These were then aggregated using the same feature engineering functions as before (mean, max, min, etc...) with and without mean subtraction. Additionally, nine networks were trained with additional steps after the convolutional layers. This included 50% drop out before a dense layer with one output and sigmoid activation to predict the relative source position in the water (0–1). As there were many hyperparameters in data pre-processing, time–smoothing and CNN setup, `rayvanve` trained 30 different classification networks and used the outputs from all of these, in addition to those generated in the previous section, for final classification.

The final classification was performed using a random forest. In training the random forests a feature elimination step was used to remove features that were not useful. Using a random forest for timing was attempted, but didn't perform well. As a result, the minimum Hellinger distance value was used as the time

Table 10: Waterfall specifications for 2D CNN's in `rayvanve` 's approach.

| **Competitor assigned ID** | **output** | **epochs** | **fine energy bin ratio** | **energy bin scaling** | **num energy bins** | **energy smoothing width (keV)** | **fine time bin (s)** | **time bin (s)** | **num time bins** | **time smoothing width (s)** |
|---|---|---|---|---|---|---|---|---|---|---|
| 49740429 | classification | 60 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.25 |
| 5378213 | classification | 60 | 0.25 | linear | 400 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 43594497 | classification | 40 | 0.25 | linear | 600 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 23892937 | classification | 60 | 0.25 | sqr | 250 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 18254272 | classification | 30 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 12414208 | classification | 60 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.50 |
| 18926302 | classification | 60 | 0.25 | linear | 400 | 1 | 0.0375 | 0.15 | 60 | 0.50 |
| 46572391 | classification | 30 | 0.25 | linear | 350 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 1683883 | classification | 60 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 2.00 |
| 5378213 | classification | 50 | 0.25 | linear | 400 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 43594497 | classification | 18 | 0.25 | linear | 600 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 12414208 | classification | 30 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 1.50 |
| 46572391 | classification | 50 | 0.25 | linear | 350 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 18254272 | classification | 60 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 23892937 | classification | 20 | 0.25 | sqr | 250 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 5378213 | classification | 30 | 0.25 | linear | 400 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 13162457 | classification | 40 | 0.25 | linear | 300 | 1 | 0.0625 | 0.25 | 60 | 0.75 |
| 12414208 | classification | 40 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.50 |
| 5378213 | classification | 18 | 0.25 | linear | 400 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 18254272 | classification | 50 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 30342994 | classification | 50 | 0.25 | linear | 200 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 46572391 | classification | 24 | 0.25 | linear | 350 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 49740429 | classification | 30 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.25 |
| 18254272 | classification | 12 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 90 | 1.00 |
| 46572391 | classification | 50 | 0.25 | linear | 350 | 1 | 0.0625 | 0.25 | 60 | 1.00 |
| 42996466 | classification | 50 | 0.25 | linear | 200 | 1 | 0.0625 | 0.25 | 60 | 0.75 |
| 43594497 | classification | 60 | 0.25 | linear | 600 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 30342994 | classification | 24 | 0.25 | linear | 200 | 1 | 0.0625 | 0.25 | 60 | 0.50 |
| 49740429 | classification | 10 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.25 |
| 49740429 | classification | 20 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 60 | 1.25 |
| 12414208 | localization | 60 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 1.50 |
| 4521908 | localization | 40 | 0.25 | linear | 250 | 1 | 0.0250 | 0.10 | 30 | 0.25 |
| 4521908 | localization | 30 | 0.25 | linear | 250 | 1 | 0.0250 | 0.10 | 30 | 0.25 |
| 12414208 | localization | 30 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 1.50 |
| 4521908 | localization | 60 | 0.25 | linear | 250 | 1 | 0.0250 | 0.10 | 60 | 0.25 |
| 12414208 | localization | 40 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 1.50 |
| 4521908 | localization | 30 | 0.25 | linear | 250 | 1 | 0.0250 | 0.10 | 60 | 0.25 |
| 12414208 | localization | 40 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 1.50 |
| 41001735 | localization | 40 | 0.25 | linear | 250 | 1 | 0.0625 | 0.25 | 30 | 0.25 |



Heuristic Approach
Spectra Waterfall
{0.0875, 0.25, 0.35, 0.7} sec int time
1keV bins (3MeV max)
Augmentation:
2D gaussian smoothing (2keV kernel with 8 different time widths)
Reduce num bins
30 second segements
Source Templates
2keV bins
Characteristic Distance:
For each energy spectra
Hellinger, Cosine, L2
Mean subtract
Feature Engineering:
max, min, mean, std, kurtosis, argmax, argmin, and snr
For each 30 sec segment
For 5/6 sources
CNN Approach
Spectra Waterfall
{0.25, 0.1, 0.15} sec int time
linearly increasing bins (3MeV max)
Augmentation:
2D gaussian smoothing (multiple kernel widths)
30 second segements
2D CNN:
Conv2D (twice) with activation, maxpool, MLP (twice)
Best with full width energy kernels
Two different final layers:
Classification: 50% Dropout, Top chosen (30 for classification, 9 for localization)
Localization (9 total):
50% Dropout, Dense, Sigmoid on 1 output
Classification (30 total):
50% Dropout, Dense, Softmax on 7 classes
6 source
1 bkg
probabilities
Mean subtract
Feature Engineering:
same as above
For all sources
Final Classification
Engineered Features:
from distance metrics
from CNN
per 30 sec segement
per source
Random Forest:
1D feature vector input
binary output
Iterate during training to eliminate noisey features
Source Det
Random Forest:
1D feature vector input
6 source probability output
Feature elimination
Source ID
Source time prediction
Source Time
Source specific thresholds for 100% TPR from training
No
Hellinger Distance (smaller is better)
Smaller than threshold?
Yes
Distance weighted average of times below threshold

Figure 80: Schematic of the fourth place algorithm from `rayvanve` .

estimation.

The competitor was not too happy with the overall approach, given the use of such a variety of methods. They note that training the CNN's with 40–60 epochs provided better results on the provisional data–set but this may have resulted in over–fitting and been detrimental to the private score. RNNs were attempted, however the results were not as good as 2D CNN's. One big missing component of this approach was cross-validation. As noted by the competitor, this was implemented out of laziness and likely cost the competitor points in over–fitting. Additionally, the source code includes many hard coded hacks due to what the competitors notes as "mistakes" which would hinder attempts to expand this approach with different hyperparameter configurations.

### 7.2.5 06-cyril.v

The competitor `cyril.v` used three 1D convolutional neural networks each with a LSTM layer in their algorithm. These networks were implemented in keras/TensorFlow and were used to detect the presence of a source, identify the temporal region of a source encounter, and finally to detect the time of closest approach. Similar to the previous competitor a moving time–window of 30 seconds was used to standardize input to the networks. The input was generated by performing a 2D histogram with 0.5 second time-bins and 79 logarithmically spaced energy bins (0–4000 keV). Additional Gaussian noise with a standard deviation of 0.001 was added during training.

The first network was used to detect and identify sources. This network included a three sequences of a 1D convolutional layer, relu activation, a dropout layer to avoid over–fitting and a max pooling layer to reduce the feature size by a factor of 2. The 1D convolutional layers used 128 kernels with a width of 5 time increments. This sequence was followed by a final LSTM layer which fed tanh activated features to 2 dense layers. The first dense layer was a source detector with a single sigmoid activated prediction. This output was thresholded at 0.95 to determine the presence of a source. The second dense layer was a source classifier with 7 softmax activated predictions for the null case and the 6 source types.

Training was performed using ten phases; the first with 100 epochs, the following 9 with 20 epochs. In each phase 80% of the training data was randomly selected while the remaining 20% was used for validation. After each phase, the model with the best validation was selected to avoid reducing model performance with further training. This approach is similar to early stopping which may have been beneficial here. The competitor also included a regularization layer during training which added Gaussian noise before the first convolution layer to limit overfitting.

The second network was designed to highlight the best 30 second time–slice to be used for source timing. This network has a single convolutional layer with 128 kernels followed by a LSTM layer and the same softmax output format as the previous network. Similarly dropout and Gaussian noise were included. This network was trained 2000 epochs and a 75/25 split was used for validation. Only source present data was used in training this network.

The final network performed source timing. This network had a similar design to the first network but included more kernels (256). Training was done again with a 75/25 split and 2000 epochs.

The networks were then used as follows. The networks were run on all sliding windows of the input sequence. If the first network registered an output of 0.95 or larger then a detection occurred. The second networks outputs were then used to identify the 30 consecutive windows having the largest sum of squared source probabilities. The timing was then computed by taking the weighted (probability to the fourth) average of the outputs of the final network on those 30 consecutive tine windows.

The approach is somewhat interesting in that different networks and training sets were used for the different tasks, however it is unclear whether such an approach is truly merited. Given the performance of the first place approach it seems that using three different networks may not be necessary. On the other hand, the implementation of non–linear energy binning combined with a 1D convolutional network and rolling time windows is quite promising for a fieldable network.

## 7.3 Introspection and perturbation of the top algorithm

### 7.3.1 Embedding Layer Kernels

The top scoring algorithm used pre–processed waterfall inputs of energy spectra over time. The 1D convolution layers were evaluated with feature kernels across all energies and 3 time bins. The first layer is called an "embedding" layer which converts the energy channels to feature values with 32 kernels. These kernels were analyzed to determine if the networks learned spectral features for the background and sources.

The kernels each consist of a scalar bias (offset) and array of weights (across spectrum energies). An example kernel is shown in Figure 82. The weights array heatmap shows prominent features around the primary photo–peak energies for I131 an Co60. These are large branching ratio peaks which are relatively isolated from other source or background features.

Next we analyzed all kernels in the ensemble (960 total) to look for common trends which each network learned. The weight vectors for each kernel were grouped using nearest–neighbor k–means clustering [107]. Through iteration we found that 20 groups resulted in visually distinct clusters.

The clusters, shown in Figure 83, show peak– and trough–like features around gamma–ray photo–peaks which confirm that the embedding layer learned spectral features without the provided source templates. That being said, the more complex, multi–peak source terms from HEU and WGPu are less prominent or missing altogether. This could be a result from the training process learning to associate the lack of the prominent peak features from other sources as an indicator for the presence of source which are more difficult to identify. Unfortunately the following 6 layers of abstract feature convolutions make it difficult to determine the precise effect of each embedding kernel on the final result.

### 7.3.2 Saliency Mapping

Another introspection technique, typically used identify the most significant image pixels which lead to a particular classification, provided insight into the regions of the waterfall which the network used to make a classification [108]. This approach creates a normalized weight (or importance) per input element called a saliency map which is a concept from computer vision capturing image pixel uniqueness or importance for

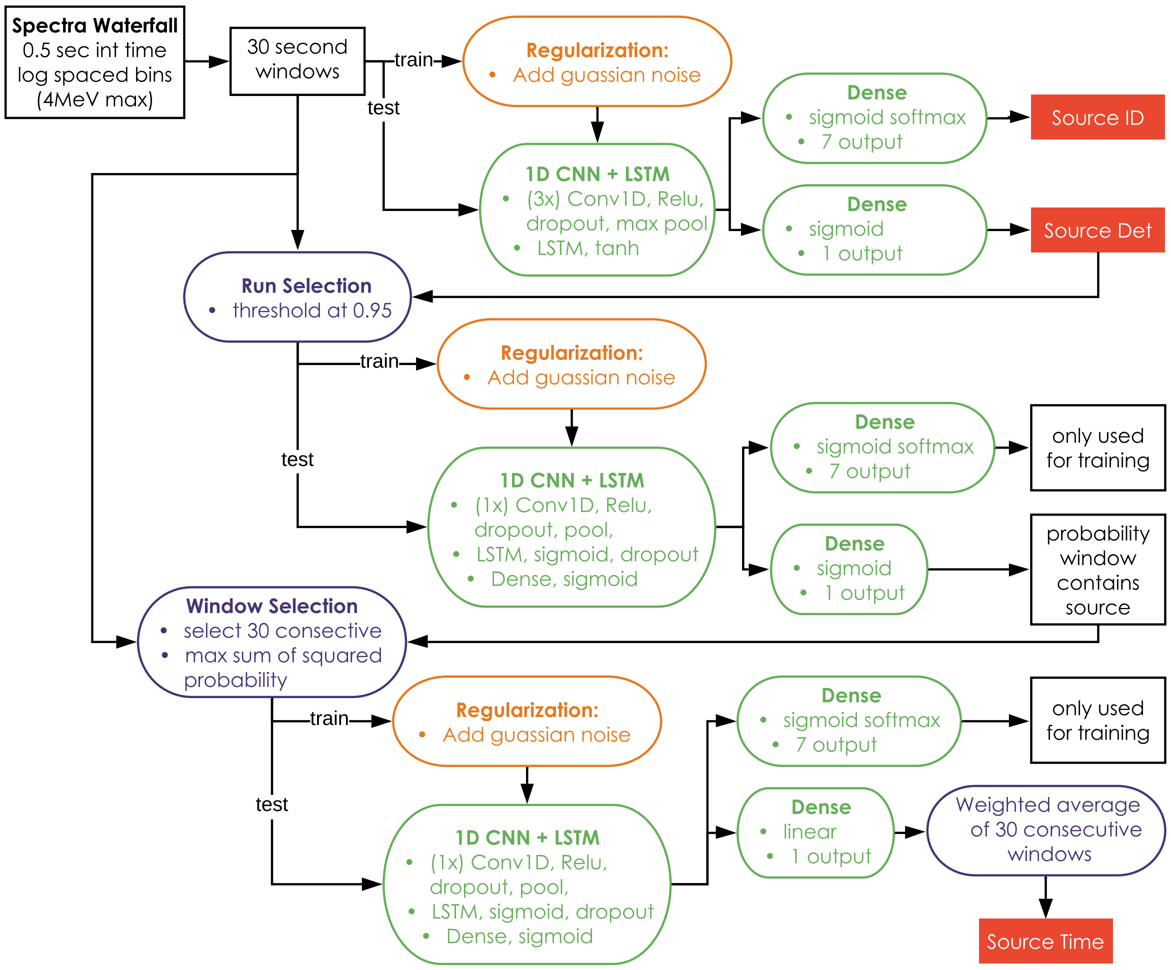


Figure 81: Schematic of the fifth place algorithm from `cyril.v` .

compression or easier analysis. We calculated the map for a particular training input with fine time bins (160 elements) without augmentation. The forward calculation through the network provides the LogSoftMax output with 1681 elements. Using the known source and time of closest approach, we selected the ground truth element and ran this element in reverse through the network to calculate the gradient of each layer with respect to this value (the ratio of the change in classification weight to the change of input value). This resulted in a final array of gradients corresponding to input waterfall elements.

An example result from this analysis is shown in Figure 84 for a $^{60}$Co source pass near time bin 135. The saliency map shows hot–spots around the 1173 and 1332 keV photo peaks which indicate that the network learned to classify $^{60}$Co using these features, much in the same way an analyst would. Additionally, the saliency reveals the time dependence of the classification result for the time of closest approach from which the map was calculated. This means that the network classified the correct source at the time of closest approach based on time windows before and after the encounter. Upon closer inspection the pre- and post-time hotspots were of slightly higher intensity than the ground truth, meaning that the network used these times equally or even more while classifying the source.

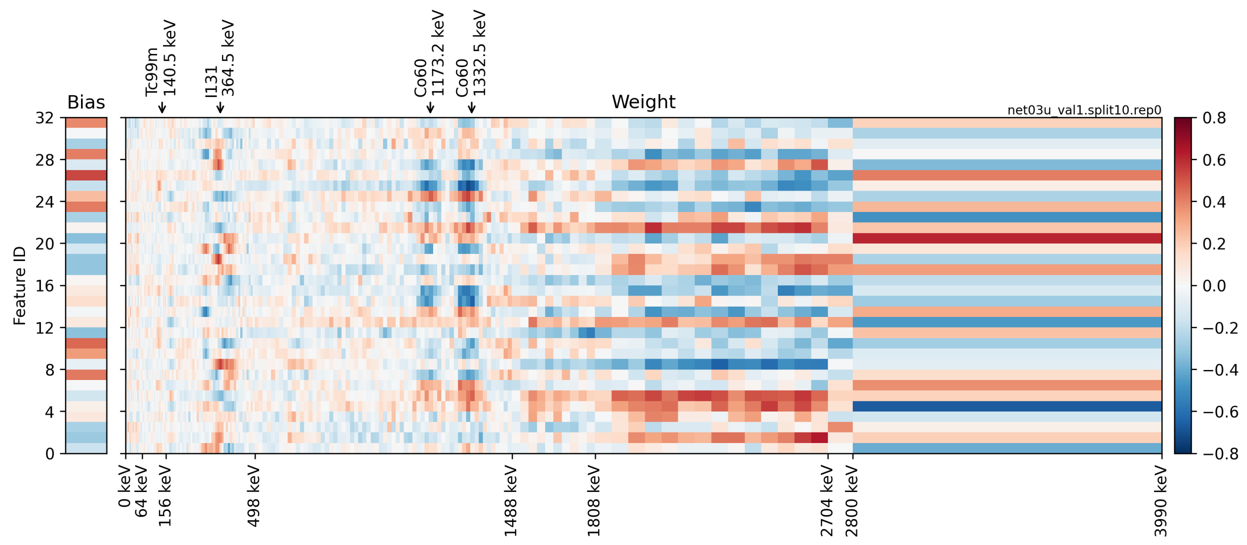


Figure 82: The embedding kernel from the first (of 30) networks in the `pfr` ensemble. Using the bias offset (left) and the weights (right), this kernel converts an energy spectrum to 32 abstract feature values. Some source peak features (I131 and Co60) can be observed in the weights array.

### 7.3.3 Hyper–parameter modifications

The algorithm structure was also modified to investigate the impact of various design choices on the final score. These parameters were defined by a flexible YAML specification which allowed algorithm modifications without substantial code refactoring. This specification included:

1. ensemble size (number of cross–validation splits and repetitions),
2. number of convolution layers (network depth), and
3. number of convolution layer output features.

In addition to these default parameters, the following parameters/customization's were added for further exploration:

1. activation function selection between convolution layers,
2. energy bin structure, and
3. training target modification.

The training target was modified to reduce the number of time bins which were set to non–zero values based on the ground–truth. This had the effect of training the network to determine a source ID at the time of closet approach instead of a surrounding time window (the default).

With these modifications, many algorithms were retrained and scored while modifying one of these parameters at a time. On a Nvidia GTX 1080-Ti the training time was between 14–18min per net such that an ensemble of 30 networks took 7.5hours in total. This training time does not include the pre–processing of the raw CSV listmode data into a binary file of 2keV binned listmode data with an auxiliary file of run indices. This CPU–bound pre–processing took almost as long as the ensemble training but only needed to be completed once for a base energy bin structure of 2keV. These results are summarized in Table 11.

The observations from this process were as follows. Changing the training target to a single time bin slightly reduced the score. This meant that while the time was inferred from the time bin with the largest response, the convolutional network picked up the rise and fall of features in the spectra which did not correlate to a delta–like output. After changing the training target, the number of time bins and network depth (number of convolution layers) were modified. Decreasing either of these parameters reduced the score

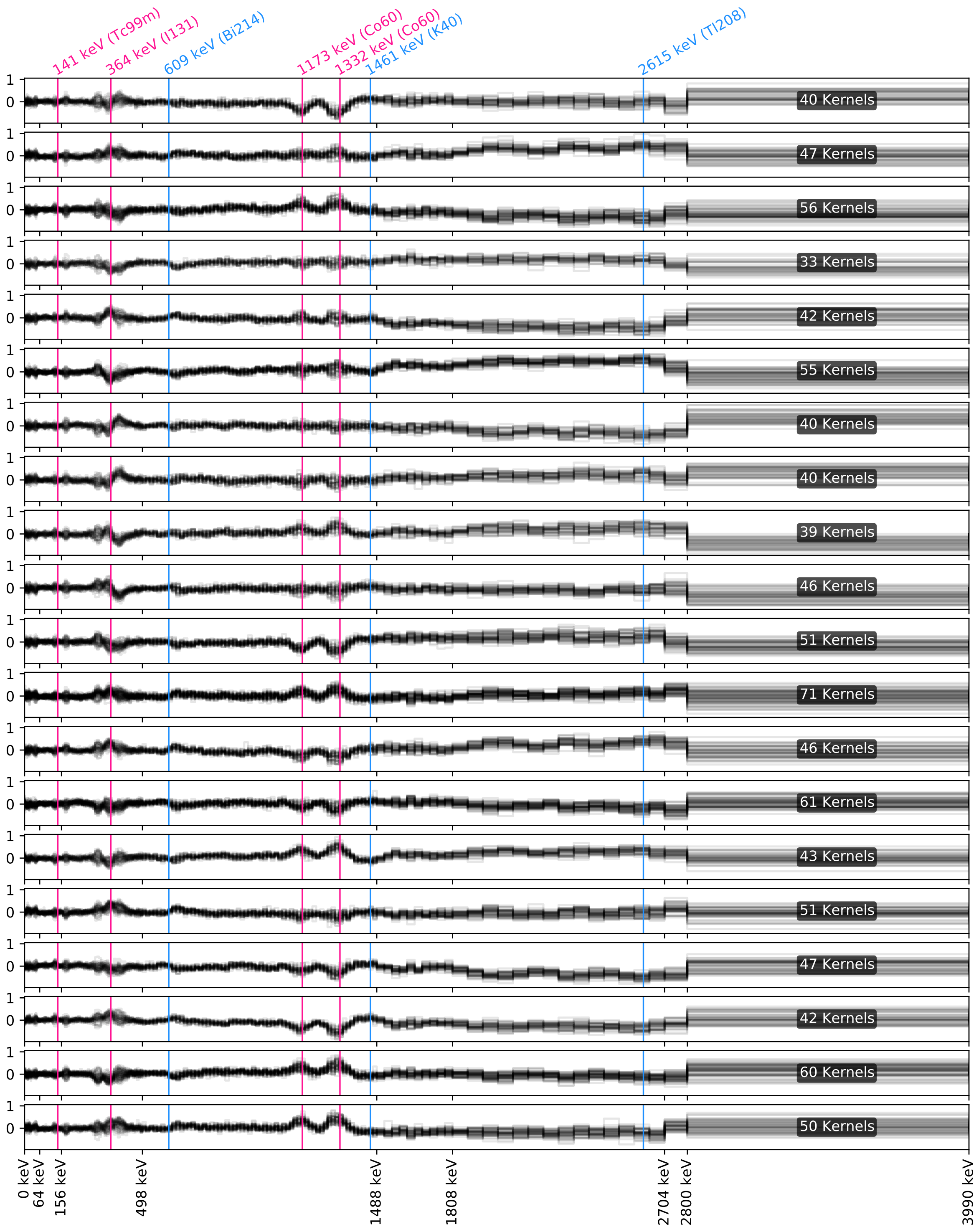


Figure 83: Nearest–neighbor k–means clusters of all embedding kernels (30 networks each with 32 embedding features) from `pfr` . The kernel clusters exhibit peak–like features around photo–peak energies of interest as marked in blue for background and pink for sources of interest.

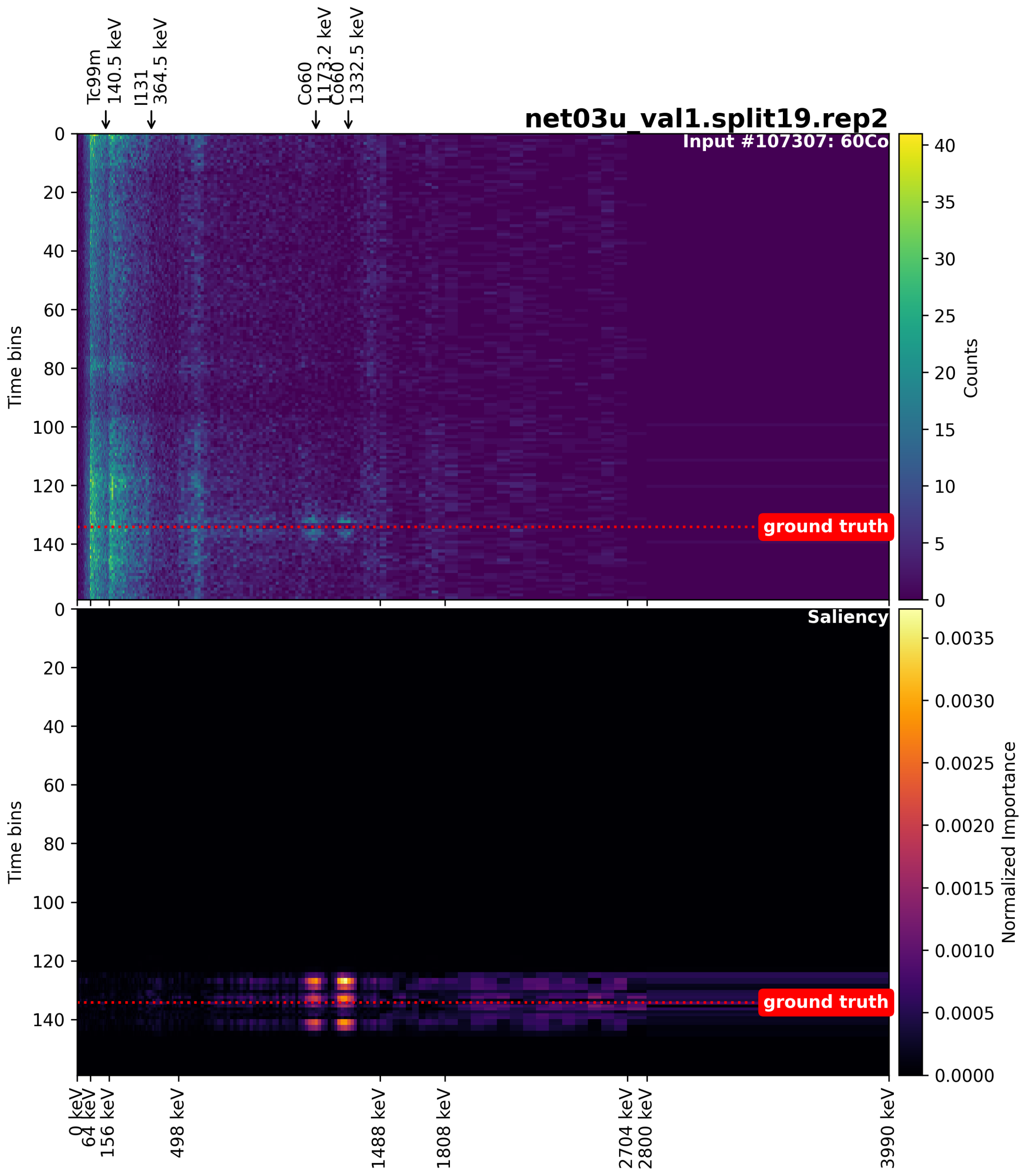


Figure 84: The saliency map of training input 107307 (source 4, $^{60}$Co) for network 30 from `pfr` . The kernel clusters exhibit peak–like features around photo–peak energies of interest as marked in blue for background and pink for sources of interest.

Table 11: Results from augmentations to `pfr` 's approach with hyper–parameter modifcations highlighted in yellow. The Embed Dim is the number of features output from the first 1D convolution layer operating on energy channels from 3 adjacent time windows. The Dim is the number of features in each sequential convolution layer. The public and private scores are colorized from red (worst) to white (neutral/baseline) to green (best).

| | Torch | Energy | Num Time | Training | | Embed | | | | CV | Public | Private |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Trial | Version | Bins | Bins | Target | Depth | Dim | Dim | Activation | Rep | Splits | Score | Score |
| 1 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 90.25 | 76.88 |
| 2 | 1.0.1.post2 | increasing | 160,80,40 | single channel | 6 | 32 | 40 | ReLu | 3 | 10 | 89.16 | 75.50 |
| 3 | 1.0.1.post2 | increasing | 160,80,40 | box car | 2 | 32 | 40 | ReLu | 3 | 10 | 89.05 | 73.86 |
| 4 | 1.0.1.post2 | increasing | 320,160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 89.89 | 76.53 |
| 5 | 1.0.1.post2 | increasing | 80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 89.56 | 75.74 |
| 6 | 1.0.1.post2 | increasing | 40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 86.68 | 73.02 |
| 7 | 1.0.1.post2 | increasing | 160,80,40 | box car | 1 | 32 | 40 | ReLu | 3 | 10 | 82.09 | 65.88 |
| 8 | 1.0.1.post2 | increasing | 160,80,40 | box car | 3 | 32 | 40 | ReLu | 3 | 10 | 89.69 | 75.99 |
| 9 | 1.0.1.post2 | increasing | 160,80,40 | box car | 4 | 32 | 40 | ReLu | 3 | 10 | 90.00 | 76.43 |
| 10 | 1.0.1.post2 | increasing | 160,80,40 | box car | 5 | 32 | 40 | ReLu | 3 | 10 | 90.14 | 76.75 |
| 11 | 1.0.1.post2 | increasing | 160,80,40 | box car | 7 | 32 | 40 | ReLu | 3 | 10 | 90.28 | 77.00 |
| 12 | 1.0.1.post2 | increasing | 160,80,40 | box car | 8 | 32 | 40 | ReLu | 3 | 10 | 90.38 | 77.01 |
| 13 | 1.0.1.post2 | increasing | 160,80,40 | box car | 9 | 32 | 40 | ReLu | 3 | 10 | 90.40 | 77.01 |
| 14 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 6 | ReLu | 3 | 10 | 86.77 | 74.15 |
| 15 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 10 | ReLu | 3 | 10 | 89.27 | 75.83 |
| 16 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 20 | ReLu | 3 | 10 | 90.03 | 76.64 |
| 17 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 30 | ReLu | 3 | 10 | 90.20 | 76.69 |
| 18 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 50 | ReLu | 3 | 10 | 90.16 | 76.83 |
| 19 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 60 | ReLu | 3 | 10 | 90.39 | 76.84 |
| 20 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 1 | 10 | 90.24 | 76.62 |
| 21 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 2 | 10 | 90.23 | 76.93 |
| 22 | 1.0.1.post2 | increasing | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 5 | 90.28 | 77.23 |
| 23 | 1.0.1.post2 | increasing | 160,80,40 | box car | 8 | 32 | 60 | ReLu | 2 | 5 | 90.26 | 76.84 |
| 24 | 1.0.1.post2 | increasing | 160,80,40 | box car | 8 | 32 | 60 | ReLu | 1 | 5 | 90.27 | 76.67 |
| 25 | 1.0.1.post2 | increasing | 160,80,40 | box car | 8 | 32 | 60 | ReLu | 3 | 4 | 90.15 | 77.08 |
| 26 | 1.0.1.post2 | increasing | 160,80,40 | box car | 8 | 60 | 60 | ReLu | 3 | 4 | 90.12 | 76.91 |
| 27 | 1.5.0+cu101 | increasing | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 90.25 | 76.92 |
| 28 | 1.5.0+cu101 | 2keV | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 89.21 | 75.33 |
| 29 | 1.5.0+cu101 | 4keV | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 89.73 | 76.02 |
| 30 | 1.5.0+cu101 | 4keV to 3MeV | 160,80,40 | box car | 6 | 32 | 40 | ReLu | 3 | 10 | 89.89 | 76.23 |
| 31 | 1.5.0+cu101 | increasing | 160,80,40 | box car | 6 | 32 | 40 | Mish | 3 | 10 | 90.33 | 77.26 |
| 32 | 1.5.0+cu101 | increasing | 160,80,40 | box car | 6 | 32 | 40 | Mish | 3 | 5 | 90.31 | 77.12 |

with the largest impact arising from reducing the depth. Increasing the number of time bins to a finer scale (320 bins) actually decreased the score. This was likely because the training only selects one integration to set a training target using a heuristic derived from the speed–to–offset ratio. The heuristic did not account for decreased counting statistics in the finer time bins so the training process may have been less effective due to statistical noise. Increasing the network depth improved the score but saturated after 7 layers which `pfr` likely tested before settling on 6 layers. Changing the number of features in the convolution layer sequence always decreased the private score leading us to believe the number of features can easily cause over–fitting to the "easy" cases. Unsurprisingly, changing the energy bin structure to a linear form decreases performance since the number of energy bins per feature very with energy. Finally, decreasing the number of cross validation splits (combinations of 90/10 training and validation sets) resulted in the largest increase in private score for a single parameter modification. In this case (trial 22), the ensemble was half the size of the winning algorithm and open up the possibility for reducing algorithm complexity (and inference time) while maintaining performance. It is worth noting that reducing the repetitions of augmentations (re–sampling and changing time direction) of the same cross–validation splits also increased the performance but not as dramatically as changing the cross-validation data altogether as in trial 22.

After the single parameter search, the best scoring modifications were mixed to reduce the network size with the goal of maintaining or improving the score. The best combination was trial 25 which improved the score with a smaller ensemble of deeper networks. This ensemble took 10% longer to train but the increase in inference time was negligible.

During this analysis the version of PyTorch (the neural network framework used by `pfr` ) was upgraded in order to leverage other tools for introspection and visualization. The upgrade slightly improved the score and was used for trial 27 and onward.

The final modification required some additions to the source code to implement a different activation function which has been show to improve objection detection and segmentation tasks [109]. This function (named Mish) is a non–monotonic function which differs from the ReLu function used by `pfr` in that it allows negative values and introduces non–linearity to the output of each convolution layer. A collection of activation functions is shown in Figure 85. Similar to the results in other domains, replacing all activation convolution layer activation functions with Mish resulted in an improvement over the baseline score. In fact, drop–in–place activation function update resulted in the largest private score improvement in this analysis as seen in trial 31. Additionally, reducing the ensemble with half the cross–validation splits resulted in a lower but still higher score than any other modification. The improvement comes with some cost since the Mish function is more complicated than the ReLu function but was not determined to be a limiting factor with 15% more training time and equivalent inference speed. Optimizations of the Mish activation function for specific hardware are currently under development and are expected to soon be available in the stable release of PyTorch.

## 7.4 Key findings and best practices

From the results of the TopCoder competition, it is clear that neural networks and random forests are capable of performing detection and identification of radiological sources with high sensitivity. This was most clearly shown by the clean and direct approaches from the 1st and 3rd place competitors. However, this point is also conveyed in the success of efforts like that of the 2nd place competitor, where many algorithms and approaches were combined to a high–scoring end. This leads us to an important point, which was known and acknowledged from the beginning of this project, the competitors objective was to maximize their score. Given the broader motivations and complexities of the radiological search problem that were not embedded in the scoring routine, or even in the data for that matter, these algorithms are not silver bullets and were never expected to be. Rather they are demonstrations that a challenging competition was held, and that the challenge was met with different approaches that outperformed the conventional baseline and the .gov competition winners.

Several of the best practices observed in the winning algorithms resonate with subject matter experts. These include histogramming gamma–ray data to standardize data format, non–linear energy binning such that bin width scales with detector resolution, analysis of multiple time–windows given the desire to optimize SNR in variable encounter speeds/standoffs, and binomial down-sampling of the gamma–ray data to generate additional data while maintaining statistical properties. Beyond these we note the data–driven learning

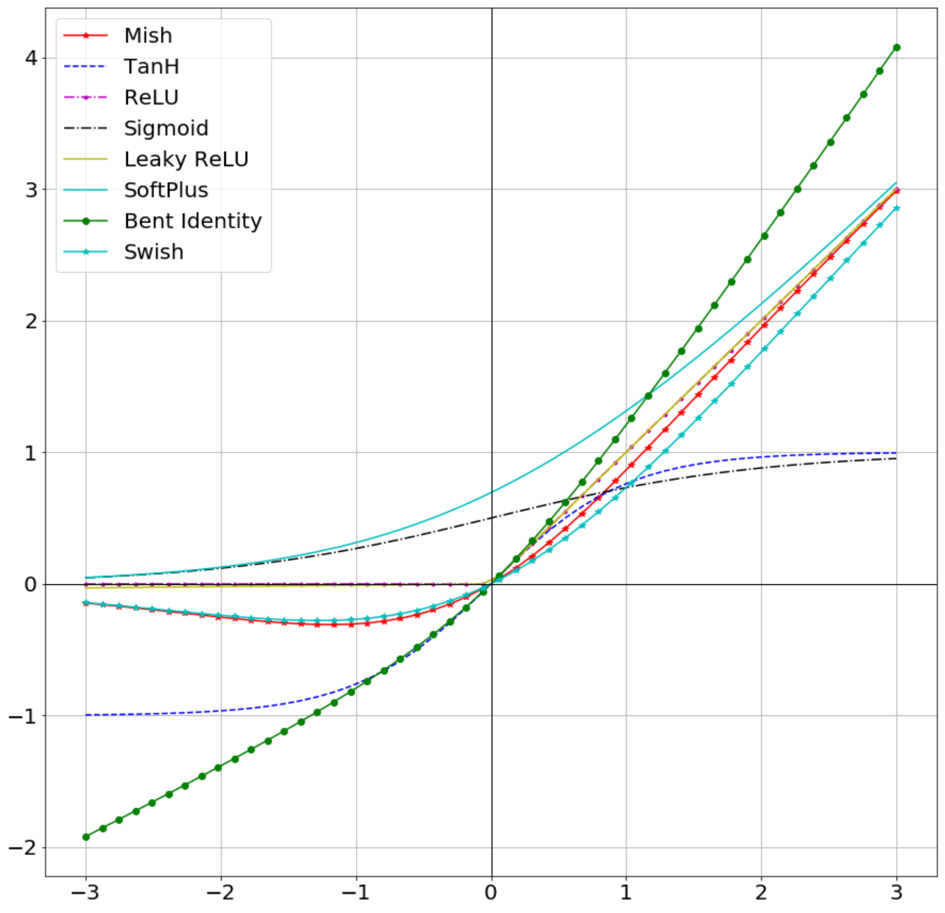


Figure 85: Typical activation functions used in neural networks. Credit: `https://github.com/digantamisra98/Mish`

employed by most competitors. By this we mean that the source templates, provided with the competition data set, were not used, rather competitors labeled their training data and used this to teach algorithms how best to identify the sources of interest. This approach has the advantage of teaching the algorithms how to identify sources in complex scattering environments typical in urban scenes which are not captured in the ideal simulated source templates. Additionally, the algorithms that analyze the time and energy information simultaneously learn not just the variable spectral signals, but how theses signals evolve during the source pass which cannot be gleaned from the source template alone. Such a practice puts the onus on diverse and complex training data, which in the case of the competition was ample, but for larger source libraries may create a burden.

Throughout the top algorithms we see common "machine learning best practices." Many of these focus on preventing over–fitting. These include using separate subsets of data for training, validating, and testing algorithms, using dropout in neural networks during training, and adding different noise to algorithm inputs. Other best practices focus on making training data go further. Examples of this are data augmentation via time–reversal or binomial down-sampling which expands the available training data. Finally, the way in which data is labeled for algorithm training is crucial. The labeling used should be consistent with how the selected algorithm computes loss such that learning will progress well. Additionally the labeling should follow simple models for signal strength (like inverse–square weighting) instead of binary labels since these outputs are generally expected to scale with source proximity.

Many competitors also engineered features from the provided data to extend the algorithm input size. These features were intended to provide additional information to the algorithms, especially in cases where the algorithm analyzed time steps sequentially and could take advantage of full run metrics like gross counts and duration. Competitors derived other features from summary statistics (mean, min, max, skew, etc) of distributions of listmode values, waterfall counts or intermediate algorithm results. While these features increased the information each algorithm could operate on, it is worth noting that the top competitor did not use this approach.

In terms of neural networks, there were two primary convolutional neural network designs seen in the top 5. The first was a 1D convolution in time, where the kernels were the width of the energy axis. The second was a 2D convolution over time and energy. While the 2D convolutions could sweep over time and energy in the 2D histogram (waterfall) `rayvanve` notes that the best performance they observed came from a convolution kernel which extended to nearly all energy bins – essentially a 1D convolution in time. This approach was initially surprising as we typically analyze spectral features with energy windows when detecting and identifying sources. After observing this approach in every CNN based algorithm, we realized that this is a close analogy to audio and other spectrograph analysis [110].

In addition to CNN's, some competitors implemented recurrent neural networks which use the previous output as features for the next input. This approach makes the inputs dependant on the previous input which can stabilize results for highly varying (low counting statistics or SNR) inputs. These approaches have some drawbacks, such as the vanishing gradient problem during training, which are typically resolved through using long short–term memory (LSTM) networks which improve the memory of previous inputs (especially during training) [111]. LSTM's are particularly suited for time sequence analyses such as analyzing a rolling window of spectra which, in theory, should improve performance during source encounters. That being said, most of these approaches were not optimized and competitors often noted that their future work would include hyper–parameter optimization of the recurrent layers. While RNN's can provide improved performance, this comes at the cost of notoriously difficult hyper–parameter configuration which some competitors (like `pfr` ) didn't attempt and relied instead on convolutions in time.

Another difference between neural network approaches included the pre–processing or standardization of the input shape prior to analysis. Neural networks operate on fix sized input (i.e. constant number of time and energy bins) which posed a challenge since the each run varied in length based on the speed of the encounter. In some cases this was a fixed rolling time–window, while the winner chose different levels of discretizations for the whole run. The fixed number of time bins over various run extent likely gave `pfr` an advantage since the many integration times resulted in additional count rate variation to improve scale invariance. One can envision translating such an approach to different fixed integration times and employing a smaller total length window for a real-time analysis, while maintaining the same receptive field of final neurons. Such an approach would maintaining sensitivity at different speeds while still providing a fixed size input.

In terms of decision trees, we observed two approaches. The first, Random Forests [103], is the simpler approach consisting of a ensemble (forest) of binary decision trees which, while training, drop features in the decision nodes to avoid over–fitting. The training the forest of trees has a similar effect to K-fold cross validation since the trees are trained independently on a subset of the training data. The objective of case of source detection is a classification task which is achieved by majority–rules voting by all trees after execution. While regular decision trees offer great interpretability, the stochastic nature of random forests leads to a more complex solution (akin to neural networks).

The second type of decision tree used by 1 competitor (and attempted by 2 others) are a relatively new approach called light–gradient boosted machines (LGBM) [104]. Gradient boosting machines (GBM) [112] combine shallow decision trees in sequence as opposed to in parallel such that each tree can improve the results from the previous tree. While GBM's offer improved anomaly detection over Random Forests, they are computationally expensive to train/execute and thus do not scale to large data sets. Light GBM's were created to address these issues through multiple approaches to reduce and bundle features such that computations are only performed on features which greatly impact the output. In the past few years LGBM's have become quite popular since they offer similar accuracy to GBM's at 20x the speed.

While LGBM's are a popular tool in the data science community, it is worth noting that they are harder to tune than RF's which, if done incorrectly, can lead to over–fitting on input with high noise. There is some evidence of this assessment in the competition results since the second place competitor used only RF's. This algorithm used RF's exclusively and thus the competitor likely focused on parameter tuning. Lower scoring competitors used LGBM's as part of a larger ensemble with NN's which they mentioned could be further tuned to boost performance but instead they used neural networks.

The top competitors implemented a variety of the neural network and/or decision tree approaches discussed above in some form of an ensemble. These collections of independent or sequential algorithms were trained on different subsets of data (cross–validation) to reduce bias towards the training data and improve accuracy for new and diverse testing data. The ensembles do take longer to train than single networks, which must be balanced with the overall robustness resulting from minimizing overfitting.

All algorithms ended with the application of a threshold to determine which of the source classes (if any) were most likely present and, if so, at which time increment was this probability the largest. In a classical algorithm, this threshold would be set by choosing a probability of detection (PD) for each source and a false alarm rate (FAR) from background noise. As the objective of the competition was maximization of the score each competitor chose thresholds based on trial and error instead of a dedicated analysis. In some cases this process was leaderboard probing while in other's they used the cross–validation results to inform the values. Many competitors noted that a more detailed analysis of a threshold could be calculated but instead focused on hyper–parameter optimization to improve the algorithm as a whole.

## 7.5 Comments about utility as deployed algorithms

In the TopCoder competition the participants had a single goal, to optimize their score. As a result, the winning algorithms have been tailored towards the competition data format and scoring methodology. Algorithms that are deployed in the field should meet several criteria, of which several are in tension with the competition format as shown in Table 12.

Table 12: A comparison of fieldable algorithm requirements vs the competitor goals.

| **A fieldable algorithm should:** | **The competition encouraged:** |
|---|---|
| Perform detection of the presence of anomalies quickly | Use as much data provided as possible |
| Perform source classification of 20-70 source types | Only 5+1 source types |
| Have the ability to disable specific source types | Focus on 5+1 source types |
| A configurable false alarm rate | Optimize sensitivity for scoring algorithm |
| Be robust to quick variations in background (rainfall, tunnels, bridges) | Learn provided background variability |
| Faster than real time analysis on available hardware | Optimize for score, no concern for run-time |
| Be trainable / operable for different detector types and sizes | Optimize for competition data |

Despite this, several of the concepts employed in these winning approaches could be adapted to better align with the needs of fieldable systems. Below we briefly address each one of these points and how one might conceptually adapt some of these algorithms to the fieldable criteria.

**Perform detection of the presence of anomalies quickly.** The competition provided all of the data for a particular segment at once, and many competitors used all or large chunks of this data as inputs to their algorithms. However, these same algorithms could be adapted to take narrower time–segments of data as inputs and to operate on a rolling basis. As an example, while `pfr` performed time–chunking of the whole run into 40, 80, 160 segments, the algorithm could be adapted to analyze 16 seconds of data with 11, 22, and 44 segments (with integration times of 2, 1 and 0.5 seconds respectively) and operated on a rolling basis. Such an adaptation would maintain the receptive field of the `pfr` network (11 time-steps of the coarsest discretization) while including sufficient data to describe the local environment for a vehicle-borne system in an urban environment.

**Perform source classification of 20–70 source types.** The competition was limited by TopCoder in the size of data set that could be provided, thus the source catalog was limited in size to ensure sufficient complexity could be provided. Given the data–driven approaches of the competitors, additional training data would be needed for each source type. This is achievable, though robust synthetic data generation tools would be needed for extension to different detector sizes and types. With additional training data, one can easily imagine extending the source classification dimension of the final layer of the neural network approaches for classification of more source types. However, a common challenge in deployed algorithms is the presence of cross–talk, or alarming on incorrect but similar source classes in the field (e.g. 99Tc triggering an HEU alarm). Inclusion of sparsity regularization on these final layers of a neural network and post–processing with a decision tree could help to limit cross–talk. An additional consideration is whether a network would need to be re–trained completely when adding sources to the catalog. By separating spectral feature estimation from temporal inference one might be able to preserve the performance of the `pfr` approach while enabling simple and lightweight extension (or reduction) of the source library.

**Have the ability to disable specific source types.** The final layers of the classification tools, neural networks or otherwise, could be modified post–training to remove classes prior to calculation of a SoftMax.

**A configurable false alarm rate.** Many of the competitors hard coded threshold values for their algorithms. These thresholds could be replaced with configurable variables that are trained with sufficient background data to yield a configurable false alarm rate.

**Be robust to fast variations in background** The TopCoder data set use a single street geometry for synthesis of the data and did not include cosmic rays, radon fluctuations, or features like bridges and tunnels that can cause rapid background transients in search systems. Additional training data could be provided to the algorithms for a data-driven mitigation strategy. Alternatively, secondary algorithms could be implemented that evaluate alarming radiological data for consistency with known background features and neglect time-dependence that often drives the false alarms of concern.

**Faster than real time analysis on available hardware.** While many of the top algorithms were quite complex, their run-times were still faster than real time. Several are implemented to require GPUs for training, however these algorithms are small enough that we expect they could run in realtime on a CPU.

**Be trainable / operable for different detector types and sizes.** Again, as these approaches are data-driven if sufficient training data is available then the algorithms may be trained, however the performance and hyper-parameter optimization may not be robust across detector regimes. The statistical properties of radiological backgrounds shift in smaller detectors, additionally much better resolution systems create additional spectroscopic features to be accounted for. One concept for partial robustness to detector performance would be to train algorithms on a per detector basis for estimated spectral deconvolution, yielding an estimated incident gamma-ray flux, then passing that flux (which is consistent across detector types) into an anomaly detection and identification algorithm. If uncertainty estimation could be produced in deconvolution then the training of the anomaly detection and identification algorithm could be trained across uncertainty regimes (relevant to different size/resolution systems).

## 7.6 A "Fieldable" Concept Based on Lessons Learned

Based on the clear performance gap seen between `pfr` and the other competitors in results of the TopCoder competition and our analysis, we consider an augmentation of the winner's approach as the route toward a "fieldable" algorithm that offers the highest possible performance. The design of the `pfr` network employed three integration times, determined by the run-length, and a single convolutional sub-network applied across the entire run. A modification of this approach would fix these integration times (e.g. 0.5, 1, 2 sec), as well as the duration of data used per analysis, e.g. 22 seconds – the minimum duration to use the full receptive field of the 6 layer convolutional network. Training of this modified approach would require little alteration to the existing algorithm, however incorporation of best practices such as random cross-validation and early stopping would reduce the risk of over-fitting. Other best practices like the data augmentation schemes used (binomial downsampling and time-reversal) would be continued. The feature depth of the network would need to be significantly increased to support a larger source library.

With these concepts the algorithm would begin to fit our initial list of requirements, however modification of the source library would mandate re-training of the entire network. To circumvent this requirement, the sub-network could be split (for at least a subset of the convolutional layers) to separate spectral and temporal feature learning. With such a modification the spectral sub-network could be trained for new source-types and appended to an existing network thus avoiding re-training. The performance of such a concept is something that could be benchmarked on the existing TopCoder data set. Finally, introspection and alarm information could be provided to users via saliency mapping as discussed in the previous section.

An alternative, or perhaps complimentary, route toward a fieldable algorithm would be pursuit of the third place (`gardn999` ) methodology employing random forest classifiers or boosted trees like LGBM. Though there was a clear performance gap between `gardn999` and `pfr` , implementation of a random forest per source is extensible, fast, and offers a reasonable degree of introspection. With a growing library of sources the problem of cross–talk is likely to become an important issue that would need to be addressed. One approach to alleviate this cross–talk could be using a secondary layer of forests to "vote" on the results from the first layer. This would maintain the extensibility of the first layer to additional source classes while retraining the second layer to maintain detection accuracy.

Using both neural network and decision tree approaches could provide the most performant algorithm, but would require the most effort to design and tune. Multiple competitors used such hybrid approaches, however most of these designs were informed by random trial and error as opposed to dedicated tuning. Knowing that these complex approaches did not outperform the relatively simple CNN from `pfr` and the decision tree only approach from `gardn999` , a simple hybrid approach could be quite effective. Starting with a CNN like that of `pfr` and analyzing the time series source class output with a random forest could provide a compelling base for fieldable algorithm.

Another option could be to create an ensemble that is a combination of multiple methods that is performed via a meta-analysis. This could potentially leverage the best of each of several different approaches and lead to overall improved performance. An initial survey of a collection of popular ensemble approaches was performed on the top three algorithms. The algorithms were run on both the training and testing datasets and the outputs were recorded in CSV file format as specified by the data competition rules. Five different ensemble techniques were investigated for combining the outputs of these three algorithms: multilayer densely connected neural networks, support vector machines with linear and nonlinear kernels, k-nearest-neighbors, traditional decision trees, random forests, and extremely randomized trees (extra-trees). For the latter 5 methods, two separate models were trained on the training data, one for classification of the source type and the other for predicting the source location (regression). These five models were trained directly on the predicted source and source locations predicted by the 3 algorithms on the training data. For the neural network, the classification decisions from the 3 algorithms were each one-hot encoded into a binary array of length 7, with each value representing a single source. These three one-hot encodings were then concatenated into a single binary array of length 21, which was used as the input to the network. The output of the network is a single dense layer with 7 units, each representing a single source type, that uses the sigmoid activation function. Networks with one and two hidden layers of various sizes (between 5 and 100 units each) were tested with several different activation functions, with relu yielding the best results. Dropout layers with dropout rates of 10% were included after each hidden layer to minimize overfitting. The networks were trained using RMSProp with the categorical crossentropy loss function. The best-performing algorithm on the training set was then selected as the final model to be evaluated on the testing set. After training and optimizing the 6 different ensemble models, they were run on the output CSVs of the algorithms on the testing set and new solution CSV files were generated. These solutions were then scored using the same scoring algorithm from the competition. These results are tabulated below in Table 13. Of these 6 models tested, the best performance was obtained by the multilayer densely connected neural network, which outperformed both the second and third place algorithms, but did not reach the performance of the first place algorithm.

Table 13: Final scores for the ensemble algorithms compared to individual performance of the top three competitors.

| **Algorithm** | **Testset Score** |
|---|---|
| pfr | 76.43 |
| kuzmin | 73.67 |
| gardnn | 73.43 |
| Multilayer Dense Neural Net | 74.29 |
| Random Forest | 73.45 |
| K-Nearest-Neighbors | 73.44 |
| Extremely Randomized Trees | 73.38 |
| Decision Trees | 72.91 |
| Support Vector Machines | 71.20 |

These results indicate that a simple, naive ensemble method is inadequate for producing a superior algorithm in this particular case. There are three likely reasons behind this. The first is that, in order for an ensemble method to yield better performance, each algorithm must have cases in which they perform better than the other algorithms, that is, each algorithm must have a unique set of strengths and weaknesses such that one algorithm's weaknesses are balanced out by another algorithm's strengths. In this particular case however, the `pfr` algorithm substantially outperforms the second and third place algorithms in the majority of cases, as is demonstrated in Chapter 6. The second potential contributor to the lower ensemble performance is the fact that the ensemble methods did increase performance on the training set, but this increase in performance did not extend to the testing set, indicating that the ensemble approach did not yield a fully generalizable algorithm. As discussed in Section 6.1.2, there were considerable differences in the location sampling between the training and testing datasets, which may make it more difficult to create an ensemble that generalizes well based on the limited amount of information in the output solution CSVs used to train the ensemble. In addition, the structure of the training and test sets was such that they emphasized different regions of the input space through the use of Non-Uniform Space Filling designs (see Section 6.2), which provides another test of generalizability.

This leads to the third potential contributor, which is the fact that there is a considerable amount of information loss when moving from the actual algorithm outputs (probability distributions per source class) to final output decisions (integer class). By using the class probability distributions from each of the top three algorithms, better performance is likely to have been achieved from the ensemble.

Overall, the reduction in final score when using the ensemble primarily arose as a result of the ensembles having a lower classification accuracy, with the best ensemble having a classification accuracy 0.71% lower than that of `pfr` and a source location bonus of 0.58% higher than that of `pfr` . Despite the increase in source location accuracy, the final score was still lower by 2.14 points.

The fact that a simple, naive ensemble approach did not produce a better algorithm indicates that more complex and sophisticated approaches to combining these algorithms needs to be investigated. A more complex ensemble approach such as boosting, which requires retraining of the models, could potentially yield a better ensemble by focusing the training process on each algorithm towards their respective strengths and weaknesses. At a higher level, creating an entirely new, single algorithm that combines the most attractive features of each of the highest-performing algorithms could potentially produce a more advanced and generalizable algorithm.

# 8 Discussion and Conclusions

## 8.1 Conclusions

The NA-22 funded a Data Science project (FY2018 – FY2020), that sought to demonstrate the capability to host data competitions to leverage expertise from the broader data science and urban radiation search communities through crowd sourcing, was successful on many fronts.

First, the team gained valuable experience by hosting a government-centric competition at `https://datacompetitions.lbl.gov/competition/1/`. It was possible to learn about the key elements of hosting a competition and the many decisions and elements that need to be considered to provide the right data with the right leaderboard scoring metrics. The second competition, hosted by TopCoder for an international community of competitors, allowed broader participation and innovative algorithms from outside of the established radiation search community to be presented. The wide participation between the two competitions provided the ability to see many alternative approaches and make direct comparisons between them. Navigating the logistical challenges of hosting the competition was non-trivial, but by successfully hosting the two competitions, there is now a documented path for how future competitions might be hosted. It was important to demonstrate the capability of the DOE complex to generate and execute a successful data competition.

New statistical methodology was developed for efficiently creating large data sets with desirable intentional characteristics for each of the training, public test and private test data sets. The structure of the data allowed analysis to extract detailed comparisons between solutions for different characteristics of the detect, identify and locate portions of the competition for each of the 6 specified sources. The data generation using the ORNL/MUSE tools was effective and may find future uses in other related applications. Additional methodology was developed for the analysis to allow for in-depth comparisons between the competitors results at the global scoring level, as well as for individual components of the competition objectives.

An added benefit of the competition has been that the data set has been more broadly disseminated after the competition and is being used by many students and researchers as a way of testing new approaches, comparing their methods to established baselines, and leveraging what was learned from the competition results.

The top approaches outperformed the winning algorithms from the initial competition (.gov) that was internal to the national laboratories. In exchange for prize money, the 10 performers from the TopCoder competition turned over their analysis code with a technical description of their algorithmic approach. Through exploration and modification of this code the team found several new insights about strategies for successful urban radiation search. Artificial neural networks and tree-based algorithms were dominant among the top teams. In addition to employing these algorithms, the top competitors followed many of the best-practices common in the data science community; separating training-validation-testing data, data augmentation, dropout and other strategies to avoid overfitting, and ensembling. Furthermore, the top competitor provided clearly documented code with simple high-level configuration (again following machine learning best practices) that enable straight-forward modification of their algorithm for further evaluation.

After exploring the winner's code, and considering the simplifications that were necessary to host a successful competition, it is clear that several of the approaches found within the algorithms may be effective if developed further. The most simple and elegant approaches were also the highest performing, and there are several clear paths toward modifying these algorithms to better suit the true urban search problem. Through experimenting with the competitors' code, it was possible to understand and further optimize some of the approaches, as well as begin the process of adapting them to more operational settings. The combination of using subject matter expertise to inform machine learning techniques shows promise for further improving solutions and refining the existing capabilities.

This data competition project also demonstrated the ability to generate a well-constructed representative data set that can be used for the development of novel methods for radiation detection and identification. This is highlighted by the recent development of two deep learning-based algorithms for radiation detection and identification. One of these is recent state-of-the-art work [72], where the dataset developed for the competition was used to pre-train a 2D CNN on gamma-ray waterfalls. The network was then retrained on data from a real detector system with labeled anomalies and proved to perform well in such an environment. The data competition dataset was also used to design, develop, and test the autoencoder radiation anomaly detection (ARAD) algorithm, funded through the Department of Homeland Security, which has been tested and proven for use in a real-world detection scenario [60, 69, 70]. These accomplishments demonstrate that a dataset with desirable characteristics is a value tool that is already having an impact in the real world. It enables new types of algorithms that can work with real detector data, and it highlights the role that large, high quality synthetic datasets can play in training these new kinds of algorithms.

We conclude this project optimistic about the promise of machine learning tools applied to the urban search problem and, more generally, about the ability for broad community-scale efforts to advance capabilities within the DOE Complex.

## 8.2 Lessons Learned

While overall the competitions were highly successful for their expressed goals, there are a number of important lessons that were learned that should guide any future data competition endeavors.

### 8.2.1 The Opportunity

As noted in the introduction, there was initially skepticism from some in the radiation detection community that data science based approaches were relevant and could be competitive for solving this problem. The competitions were able to demonstrate that just by extracting features from just the data could be sufficient for detecting, identifying and locating sources from an urban environment. The direct comparison of performance from the different solutions allowed us to gain some understanding about the strengths and weaknesses of these approaches.

In addition, some of the top solutions used a combination of domain knowledge to set up the data processing and then data science based methods for the analysis. This was a powerful combination with potential to outperform alternative approaches. Without the competition to encourage data science involvement and to provide a formal comparison mechanism, it would have been much more challenging to demonstrate the relevance and potential of data science methods in this problem space.

The existence of the dataset has been noted by the urban radiation search community, and it has become a valuable tool for early assessment and development of innovative algorithms. Not only is it valuable to the development community, but results from early testing provides direct comparison of new methods to existing approaches to help gauge their potential.

### 8.2.2 Data Construction

The choices of which data to present to the competitors and how to score the results are two of the most important decisions when fielding the competition. Overall, we think that the two competitions reflect our detailed commitment to making these choices with input and guidance from subject matter experts.

1. The size of the data set is bounded above by how much data competitors can easily handle. It is bounded below by considerations of adequately exploring the input space of interest and the power to assess differences between algorithms. The upper bound is likely to change steadily over time as users are increasingly able to handle larger data sets. The lower bound of how much data is needed for each case (here each source), should be thoughtfully considered using statistical power analysis to anticipate the ability to find statistically significant differences between competitor algorithms for different terms in the model.

2. Care should be taken to identify different input factors that will be explored in the data set, and the ranges of each factor should be evaluated based on what is likely in real world scenarios and to include both regions where success and failure are likely. It is also helpful to include some “very challenging” cases to encourage algorithm growth and to ensure a demanding test for the competitors.

3. The structure of our data sets were to make the training set contain a larger number of easier cases, then progressively increase the emphasis on difficult regions of the input space for the public and private test sets. This encouraged the competitors to demonstrate the ability of their algorithms to more challenging extrapolation cases. Algorithms that can do this well, are more likely to perform well in new scenarios not directly tested by the data competition.

4. One consequence of the differentiated data sets for the training and test sets was that there were a smaller number of runs for what were considered easier scenarios. This resulted in some instability and unintuitive logistic model predictions based on the larger variances associated with some of the easier regions of the input space. In future, it would be desirable to increase the proportion of runs in these easier regions for the test set data to add some stability to the model predictions.

5. Another aspect of the data set construction that would have been beneficial to manage more actively was the balance of runs for the different source locations. There were some considerable imbalance between locations across the different training and test sets, that had some less desirable consequences. In future, it would likely be beneficial to look to manage the choice of runs to maintain some rough balance between levels of the inputs to ensure that no undesirable artifacts are introduced.

6. Thorough testing and checking of the data set is critical to eliminate any unintended artifacts in the data. This is a choice that is very difficult to correct once the competition has started.

### 8.2.3 Leaderboard Scoring

Between the two competitions, changes were made to the scoring formula to reflect what was learned from the first competition. The specification of the leaderboard scoring is a critical decision that has substantial impacts on the results of the competition and how much different objectives are prioritized.

1. In preparation to field the two competitions, there was extensive alpha and beta testing of the submission process and the scoring choices. This lead to some modifications before the competition that were critical to the success of the results. Since it is difficult (or impossible) to change the scoring algorithm once the competition is started, this pre-testing is critical to increase the probability of useful results.

2. Competition's scoring resulted in participants marginalizing over a relevant variable, the false alarm rate. This was a known compromise with this competition but it is important to remember that optimizing the scoring function is the only criteria of interest to them. This feature of a fixed leaderboard scoreboard was necessary, but is in sharp contrast to the approaches common in the urban radiation search community where the false alarm rate is generally a tune-able parameter.

3. In the government-centric competition, feedback was provided to the competitors about their performance on the different objectives (by source, and by detect, identify and locate). If the goal of the competition is to drive improvement, then providing this feedback part way through the competition can be beneficial.

4. Several of the competitors ended up with scores that were quite close to each other in the final rankings. To better understand whether these scores should be thought of as effectively equivalent, or with genuine differences, new statistical methodology was developed. This helped to provide insights about how close the performance was across other similar data sets and also allowed for experimentation with different scoring mechanisms and different proportions of data from the various sources.

5. Performance of the competitors was closely tied to the number of choices that they had for each run. Some of these constraints were dictated by our ability to generate data and the overall size of data that could be effectively provided to the competitors. The structure of the competitions dictated that there was at most one source per run, and there were only 6 possible sources from which to choose. These place considerable restrictions on how to interpret results and what we are able to predict about the performance of the algorithms in less restrictive scenarios.

### 8.2.4 Specification of Requirements for Submission

One of the primary motivations of this competition was to gain insights on the urban search gamma–ray detection problem when modern machine learning approaches are applied by competitors without extensive domain–specific knowledge. In order to learn the most from these submissions the competitors were required to submit a functional Dockerfile with test/train scripts with the hope that this functional development environment would provide insight on the approach and extensibility. The competitors were also asked to submit a report summarizing their approach and implementation.

However no other requirements where placed on the requirements for prizes so the quality of the submitted reports and code varied highly between competitors. A few competitors submitted extensive reports with detailed commentary on the various data preparation steps and algorithm components with supporting diagrams, while for many others what was submitted was less structured unpolished scripts with few comments and lots of unused code.

1. Provide a pre-formatted report to participants to guide the structure of their summary to ensure that they are consistent and hit all important relevant points.

2. Require algorithm configuration files for analyses so that perturbation studies are possible (This would also be better practice for participants!). There are lots of open source tools to enable this (MLFlow, hydra etc...).

## 8.3 Future R&D Recommendations

Because of the overwhelming positive response to the data set being available for students and researchers, it would be beneficial to develop other data sets that expand on this work, broaden the scope over which data are available, and allow testing and comparison of algorithms on new sets of conditions.

Based on what we learned from hosting the two competitions, a number of improvements and enhancements have been suggested. In addition to the ideas obtained from the competitors' feedback from the first government competition, the team has proposed a number of ideas to increase the realism of future data sets. Here are a number of the priorities that would be desirable to explore as we proceed with future development:

1. Expanding the number of sources and shielding conditions would provide greater challenges for the algorithms and increase realism.

2. Explore anomaly detection and identification with additional information. For example, there might be opportunities to include scene data, historical or geo-spatial information.

3. It would be helpful to relax the structure of the data to move away from a run with limited options (just zero or one source per run). This would allow data to be in more operationally-relevant larger blocks of time with more flexibility to have 0, 1, 2, or more sources per block of time. Scoring would need to be adapted to reflect these broader sets of scenarios, where false positives and false negatives would occur in multiple times in the data block.

4. With reduced pressure to control the overall size of the data set, it would be possible to generate larger data sets with more sources, the addition of clutter (cars, people on detector path) and inclusion of radon (rain) and cosmic backgrounds. This greater diversity of conditions would provide a more realistic test on which the algorithms could be evaluated.

5. Develop a website with curated data sets and expanded performance metrics to allow anyone interested to quickly evaluate feasibility of their approach. While previously with a data competition, there needed to be a single summary score for each algorithm, we plan to expand the summaries for each submission to allow for more detailed information to be communicated about the strengths and weaknesses of the algorithm. For example, in addition to an overall score, we would provide at least 19 summaries for each submission (6 sources x detect, identify, locate + false positive rate).

6. To encourage collaboration and sequential improvement of algorithms, it would be beneficial to include open-source implementations of published algorithms for researchers to used and as benchmarking.

7. Ideally, it would be helpful to present data in a more real-time format that allows for sequential processing of the data. Instead of the results being reported after the run is completed, algorithms would be encouraged to report a source as soon after the detector has passed it. In the competition format, where the data are all available at the start of the competition, this was not a realistic implementation

that could be achieved as competitors would have all of the data available for each submission. However, with additional flexibility in presentation of data, this might be achievable and would encourage more realistic operationally-desirable implementations.

8. It would be interesting to consider generative data sets to prevent the size of the data presented to the competitors from getting too large. This could also allow for incorporating different detector response functions to really make a robust tool usable for many years to come.

9. Facilitate the development of introspection tooling along with algorithms. Such co-development and implementation will enable not just performance evaluation via metrics but also some degree of interpretation to accompany classification results. For example, it would be helpful to have each algorithm report their estimated confidence in classification, and then as part of the evaluation of the algorithms compare it with true uncertainty. By encouraging algorithms to report not only their findings, but also to provide some quantification, this would make the results more actionable in operational settings. This would allow for evaluation of the "calibration" of algorithms. This suggests additional metrics to compliment FAR & MDA.

10. In addition to considering improvements to the structure and nature of the data set, it is also beneficial to look to leverage what was learned from the competitors' algorithms. This could allow for new and improved neural network detection / identification algorithms to be developed that are structured to enable general versus architecture designed for robustness and generality.

11. Enhancing the ensemble algorithm approach to leverage the best features of multiple approaches could lead to overall improved performance.

## 9 References

## References

[1] T. Harford, Adapt: Why Success Always Starts with Failure, Picador Paper, 2012.

[2] F. Johansson, The Medici Effect, Harvard Business Press, 2004.

[3] D. E. Peplow, Monte carlo shielding analysis capabilities with mavric, Nuclear Technology 174 (2) (2011) 289–313.

[4] R. A. L. W. A. Wieselquist, E. M. A. Jessee, SCALE Code System, Version 6.2.4, Tech. rep., Oak Ridge National Laboratory, Oak Ridge, TN ORNL/TM-2005/39 (2020).

[5] C. M. Anderson-Cook, L. Lu, K. L. Myers, K. R. Quinlan, N. Pawley, Improved learning from data competitions through strategic design of training and test data sets, Quality Engineering 31 (4) (2019) 564–580. doi:10.1080/08982112.2019.1572186.
URL https://doi.org/10.1080/08982112.2019.1572186

[6] C. M. Anderson-Cook, K. L. Myers, L. Lu, M. L. Fugate, K. R. Quinlan, N. Pawley, Data competitions: Getting more from a strategic design and post-competition analysis, Statistical Analysis and Data Mining 12 (2019) 271–289.

[7] G. Takoudis, S. Xanthos, A. Clouvas, C. Potiriadis, Determining minimum alarm activities of orphan sources in scrap loads; monte carlo simulations, validated with measurements, Nucl. Instr. Meth. Phys. Res. Sec. A: Accel. Spectrom., Detectors Assoc. Equip 614 (2010) 57–67.

[8] R. Jeffcoat, S. Salaymeh, A comparison of gadras-simulated and measured gamma-ray spectra savannah river national laboratory, Tech. rep., Sandia National Laboratories, Albuquerque, New Mexico (2010).

[9] R. C. Runkle, T. M. Mercier, K. K. Anderson, D. K. Carlson, Point source detection and characterization for vehicle radiation portal monitors, IEEE TNS 52 (6) (2005) 3020–3025.

[10] R. Arlt, K. Baird, J. Blackadar, C. Blessenger, D. Blumenthal, P. Chiaro, K. Frame, E. Mark, M. Mayorov, M. Milovidov, R. York, Semi-empirical approach for performance evaluation of radionuclide identifiers, IEEE Nucl. Sci. Symp. (2009) 990–994.

[11] G. F. Knoll, Radiation detection and measurement, Wiley, 2010.

[12] W. R. Leo, Techniques for Nuclear and Particle Physics Experiments, Springer, 1994.

[13] R. D. Evans, The Atomic Nucleus, McGraw-Hill, 1955.

[14] T. Hjerpe, R. R. Finck, C. Samuelsson, Statistical Data Evaluation in Mobile Gamma Spectrometry: An Optimization of On-line Search Strategies in the Scenario of Lost Point Sources, Health Physics 80 (6) (2001) 563–570.
URL https://journals.lww.com/health-physics/Abstract/2001/06000/STATISTICAL_DATA_EVALUATION_IN_MOBILE_GAMMA.6.aspx

[15] K. P. Ziock, W. H. Goldstein, The Lost Source, Varying Backgrounds and Why Bigger May Not Be Better, in: AIP Conference Proceedings, Vol. 632, AIP Publishing, 2002, pp. 60–70. doi:10.1063/1.1513955.
URL http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.1513955

[16] H. K. Aage, U. Korsbech, Search for lost or orphan radioactive sources based on NaI gamma spectrometry, Applied Radiation and Isotopes 58 (1) (2003) 103–113. doi:10.1016/S0969-8043(02)00222-1.
URL http://www.sciencedirect.com/science/article/pii/S0969804302002221

[17] IAEA Safeguards Glossary, no. 3 in International Nuclear Verification Series, International Atomic Energy Agency, Vienna, 2003.
URL `https://www.iaea.org/publications/6663/iaea-safeguards-glossary`

[18] G. A. Sandness, J. E. Schweppe, W. K. Hensley, J. D. Borgardt, A. L. Mitchell, Accurate Modeling of the Terrestrial Gamma-Ray Background for Homeland Security Applications, in: 2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC), 2009, pp. 126–133. `doi:10.1109/NSSMIC.2009.5401843`.

[19] K. D. Jarman, R. C. Runkle, K. K. Anderson, D. M. Pfund, A comparison of simple algorithms for gamma-ray spectrometers in radioactive source search applications, Applied Radiation and Isotopes 66 (3) (2008) 362–371. `doi:10.1016/j.apradiso.2007.09.010`.
URL `http://www.sciencedirect.com/science/article/pii/S0969804307002886`

[20] D. E. Archer, D. E. Hornback, J. O. Johnson, T. M. Miller, A. D. Nicholson, B. W. Patton, D. E. Peplow, B. Ayaz-Maierhafer, Systematic assessment of neutron and gamma backgrounds relevant to operational modeling and detection technology implementation, Tech. Rep. ORNL/TM-2014/687, Oak Ridge National Laboratory, Oak Ridge, Tennessee (2015). `doi:10.2172/1185844`.

[21] S. Labov, L. Pleasance, W. Sokkappa, W. Craig, G. Chapline, M. Frank, J. Gronberg, J. Jernigan, S. Johnson, J. Kammeraad, D. Lange, A. Meyer, K. Nelson, B. Pohl, D. Wright, R. Wurtz, Foundations for Improvements to Passive Detection Systems - Final Report, UCRL-TR-207129`doi:10.2172/15011571`.

[22] T. Hjerpe, C. Samuelsson, A Comparison Between Gross and Net Count Methods When Searching for Orphan Radioactive Sources, Health Physics 84 (2) (2003) 203–211.
URL `https://journals.lww.com/health-physics/Fulltext/2003/02000/A_COMPARISON_BETWEEN_GROSS_AND_NET_COUNT_MRTHODS.8.aspx`

[23] R. Runkle, T. Mercier, K. Anderson, D. Carlson, Point source detection and characterization for vehicle radiation portal monitors, IEEE Transactions on Nuclear Science 52 (6) (2005) 3020–3025. `doi:10.1109/TNS.2005.862910`.

[24] R. Wurtz, K.-P. Ziock, L. Fabris, R. Graham, Comparing Imaging and Non-Imaging Techniques for Reducing Background Clutter and Resolving Distant Point Sources, in: 2005 IEEE Nuclear Science Symposium Conference Record, Vol. 1, 2005, pp. 338–342. `doi:10.1109/NSSMIC.2005.1596266`.

[25] D. VanDerwerken, M. Millett, R. Whitlock, Improved Detection of a Mobile Radiological Source by Means of a Temporal Radiation Profile, IEEE Transactions on Nuclear Science 66 (9) (2019) 2080–2087. `doi:10.1109/TNS.2019.2929121`.

[26] K.-P. Ziock, W. Craig, L. Fabris, R. Lanza, S. Gallagher, B. K. P. Horn, N. Madden, Large Area Imaging Detector for Long-Range, Passive Detection of Fissile Material, in: 2003 IEEE Nuclear Science Symposium Conference Record, Vol. 2, 2003, pp. 1001–1005 Vol.2. `doi:10.1109/NSSMIC.2003.1351863`.

[27] K.-P. Ziock, J. Collins, L. Fabris, S. Gallagher, B. K. P. Horn, R. Lanza, N. Madden, Source-Search Sensitivity of a Large-Area, Coded-Aperture, Gamma-Ray Imager, IEEE Transactions on Nuclear Science 53 (3) (2006) 1614–1621. `doi:10.1109/TNS.2006.875285`.

[28] D. Stephens, R. Runkle, D. Carlson, A. Peurrung, A. Seifert, C. Wyatt, Induced temporal signatures for point-source detection, IEEE Transactions on Nuclear Science 52 (5) (2005) 1712–1715, conference Name: IEEE Transactions on Nuclear Science. `doi:10.1109/TNS.2005.856905`.

[29] T. Aucott, M. Bandstra, V. Negut, J. Curtis, D. Chivers, K. Vetter, Effects of Background on Gamma-Ray Detection for Mobile Spectroscopy and Imaging Systems, IEEE Transactions on Nuclear Science 61 (2) (2014) 985–991. `doi:10.1109/TNS.2014.2306998`.

[30] K. Miller, A. Dubrawski, Gamma-Ray Source Detection With Small Sensors, IEEE Transactions on Nuclear Science 65 (4) (2018) 1047–1058. `doi:10.1109/TNS.2018.2811049`.

[31] P. Tandon, P. Huggins, R. Maclachlan, A. Dubrawski, K. Nelson, S. Labov, Detection of radioactive sources in urban scenes using Bayesian Aggregation of data from mobile spectrometers, Information Systems 57 (2016) 195–206. doi:10.1016/j.is.2015.10.006.
URL https://www.sciencedirect.com/science/article/pii/S0306437915001866

[32] R. C. Runkle, M. F. Tardiff, K. K. Anderson, D. K. Carlson, L. E. Smith, Analysis of Spectroscopic Radiation Portal Monitor Data Using Principal Components Analysis, IEEE Transactions on Nuclear Science 53 (3) (2006) 1418–1423. doi:10.1109/TNS.2006.874883.

[33] D. Boardman, M. Reinhard, A. Flynn, Principal Component Analysis of Gamma-Ray Spectra for Radiation Portal Monitors, IEEE Transactions on Nuclear Science 59 (1) (2012) 154–160. doi:10.1109/TNS.2011.2179313.

[34] T. Burr, M. Hamada, Radio-Isotope Identification Algorithms for NaI $\gamma$ Spectra, Algorithms 2 (1) (2009) 339–360, number: 1 Publisher: Molecular Diversity Preservation International. doi:10.3390/a2010339.
URL https://www.mdpi.com/1999-4893/2/1/339

[35] D. Boardman, A. Flynn, A Gamma-Ray Identification Algorithm Based on Fisher Linear Discriminant Analysis, IEEE Transactions on Nuclear Science 60 (1) (2013) 270–277. doi:10.1109/TNS.2012.2226472.

[36] D. Boardman, A. Flynn, Performance of a Fisher Linear Discriminant Analysis Gamma-Ray Identification Algorithm, IEEE Transactions on Nuclear Science 60 (2) (2013) 482–489, conference Name: IEEE Transactions on Nuclear Science. doi:10.1109/TNS.2012.2225445.

[37] D. K. Fagan, S. M. Robinson, R. C. Runkle, Statistical methods applied to gamma-ray spectroscopy algorithms in nuclear security missions, Applied Radiation and Isotopes 70 (10) (2012) 2428–2439. doi:10.1016/j.apradiso.2012.06.016.
URL http://www.sciencedirect.com/science/article/pii/S0969804312003818

[38] J. Hovgaard, R. Grasty, Reducing Statistical Noise in Airborne Gamma-Ray Data Through Spectral Component Analysis, in: Proceedings of Exploration 97, 1997, pp. 753–764.

[39] J. Hovgaard, Airborne gamma-ray spectrometry, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark, 1998.

[40] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791. doi:10.1038/44565.
URL http://www.nature.com/nature/journal/v401/n6755/abs/401788a0.html

[41] D. D. Lee, H. S. Seung, Algorithms for Non-negative Matrix Factorization, in: T. K. Leen, T. G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 556–562.
URL http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[42] M. L. Koudelka, D. J. Dorsey, A Modular NMF Matching Algorithm for Radiation Spectra, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 284–289, iSSN: 2160-7516. doi:10.1109/CVPRW.2016.42.

[43] K. J. Bilton, T. H. Joshi, M. S. Bandstra, J. C. Curtis, B. J. Quiter, R. J. Cooper, K. Vetter, Non-negative Matrix Factorization of Gamma-Ray Spectra for Background Modeling, Detection, and Source Identification, IEEE Transactions on Nuclear Science 66 (5) (2019) 827–837. doi:10.1109/TNS.2019.2907267.

[44] K. N. Shokhirev, B. R. Cosofret, M. King, B. Harris, C. Zhang, D. Masi, Enhanced detection and identification of radiological threats in cluttered environments, in: 2012 IEEE Conference on Technologies for Homeland Security (HST), 2012, pp. 502–506.

[45] B. R. Cosofret, K. Shokhirev, P. Mulhall, D. Payne, B. Harris, Utilization of advanced clutter suppression algorithms for improved standoff detection and identification of radionuclide threats, in: Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XV, Vol. 9073, International Society for Optics and Photonics, 2014, p. 907316.

[46] T. H. Joshi, R. J. Cooper, J. Curtis, M. Bandstra, B. R. Cosofret, K. Shokhirev, D. Konno, A comparison of the detection sensitivity of the poisson clutter split and region of interest algorithms on the radmap mobile system, IEEE Transactions on Nuclear Science 63 (2) (2016) 1218–1226.

[47] K.-s. Chan, J. Li, W. Eichinger, E.-W. Bai, A distribution-free test for anomalous gamma-ray spectra, Radiation Measurements 63 (2014) 18–25. `doi:10.1016/j.radmeas.2014.02.003`.
URL `http://www.sciencedirect.com/science/article/pii/S1350448714000225`

[48] O. H. M. Padilla, A. Athey, A. Reinhart, J. G. Scott, Sequential Nonparametric Tests for a Change in Distribution: An Application to Detecting Radiological Anomalies, Journal of the American Statistical Association 114 (526) (2019) 514–528. `doi:10.1080/01621459.2018.1476245`.
URL `https://doi.org/10.1080/01621459.2018.1476245`

[49] A. Reinhart, V. Ventura, A. Athey, Detecting changes in maps of gamma spectra with Kolmogorov–Smirnov tests, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 802 (2015) 31–37. `doi:10.1016/j.nima.2015.09.002`.
URL `http://www.sciencedirect.com/science/article/pii/S0168900215010517`

[50] A. J. Cresswell, D. C. W. Sanderson, The use of difference spectra with a filtered rolling average background in mobile gamma spectrometry measurements, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 607 (3) (2009) 685–694. `doi:10.1016/j.nima.2009.06.001`.
URL `http://www.sciencedirect.com/science/article/pii/S0168900209011759`

[51] J. Ely, R. Kouzes, B. Geelhood, J. Schweppe, R. Warner, Discrimination of naturally occurring radioactive material in plastic scintillator material, IEEE Transactions on Nuclear Science 51 (4) (2004) 1672–1676, conference Name: IEEE Transactions on Nuclear Science. `doi:10.1109/TNS.2004.832286`.

[52] J. Ely, R. Kouzes, J. Schweppe, E. Siciliano, D. Strachan, D. Weier, The use of energy windowing to discriminate SNM from NORM in radiation portal monitors, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 560 (2) (2006) 373–387. `doi:10.1016/j.nima.2006.01.053`.
URL `http://www.sciencedirect.com/science/article/pii/S0168900206001057`

[53] D. Pfund, R. Runkle, K. Anderson, K. Jarman, Examination of Count-Starved Gamma Spectra Using the Method of Spectral Comparison Ratios, IEEE Transactions on Nuclear Science 54 (4) (2007) 1232–1238. `doi:10.1109/TNS.2007.901202`.

[54] K. K. Anderson, K. D. Jarman, M. L. Mann, D. M. Pfund, R. C. Runkle, Discriminating nuclear threats from benign sources in gamma-ray spectra using a spectral comparison ratio method, Journal of Radioanalytical and Nuclear Chemistry 276 (3) (2008) 713–718. `doi:10.1007/s10967-008-0622-x`.
URL `http://link.springer.com/article/10.1007/s10967-008-0622-x`

[55] D. Pfund, K. Jarman, B. Milbrath, S. Kiff, D. Sidor, Low Count Anomaly Detection at Large Standoff Distances, IEEE Transactions on Nuclear Science 57 (1) (2010) 309–316. `doi:10.1109/TNS.2009.2035805`.

[56] D. M. Pfund, K. K. Anderson, R. S. Detwiler, K. D. Jarman, B. S. McDonald, B. D. Milbrath, M. J. Myjak, N. C. Paradis, S. M. Robinson, M. L. Woodring, Improvements in the method of radiation anomaly detection by spectral comparison ratios, Applied Radiation and Isotopes 110 (2016) 174–182. `doi:10.1016/j.apradiso.2015.12.063`.
URL `http://www.sciencedirect.com/science/article/pii/S0969804315304024`

[57] R. S. Detwiler, D. M. Pfund, M. J. Myjak, J. A. Kulisek, C. E. Seifert, Spectral anomaly methods for aerial detection using KUT nuisance rejection, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 784 (2015) 339–345. doi:10.1016/j.nima.2015.01.040.
URL http://www.sciencedirect.com/science/article/pii/S0168900215000716

[58] S. M. Robinson, S. E. Bender, E. L. Flumerfelt, C. A. LoPresti, M. L. Woodring, Time Series Evaluation of Radiation Portal Monitor Data for Point Source Detection, IEEE Transactions on Nuclear Science 56 (6) (2009) 3688–3693, conference Name: IEEE Transactions on Nuclear Science. doi:10.1109/TNS.2009.2034372.

[59] M. Monterial, K. E. Nelson, S. E. Labov, S. Sangiorgio, Benchmarking Algorithm for Radio Nuclide Identification (BARNI) Literature Review, Tech. Rep. LLNL-SR-767242, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States) (Feb. 2019). doi:10.2172/1544518.
URL https://www.osti.gov/biblio/1544518

[60] J. M. Ghawaly, A datacentric algorithm for gamma-ray radiation anomaly detection in unknown background environments, Ph.D. thesis, The University of Tennessee, Knoxville, Department of Nuclear Engineering, Tickle College of Engineering, Nuclear Engineering Building, 1412 Circle Drive, Knoxville TN 37996-1410, in press (2020).

[61] D. Archer, M. Bandstra, D. Bolme, S. Cleveland, J. Curtis, G. Davidson, I. Garishvili, J. Johnson, A. Mikkilineni, M. McLean, A. Nicholson, D. Peplow, A. Plionis, M. Poska, B. Quiter, W. Ray, A. Row, I. Stewart, M. Swinney, M. Willis, Mid-Project Report for the Modeling Urban Scenarios and Experiments (MUSE) Project, Tech. Rep. ORNL/TM-2018/964, Oak Ridge National Laboratory (August 2018).

[62] M. W. Swinney, D. E. Peplow, B. W. Patton, A. D. Nicholson, D. E. Archer, M. J. Willis, A methodology for determining the concentration of naturally occurring radioactive materials in an urban environment, Nuclear Technology 203 (3) (2018) 325–335.

[63] J. G. Ingersoll, A survey of radionuclide contents and radon emanation rates in building materials used in the US, Health Physics 45 (2) (1983) 363–368.

[64] S. Croft, I. Hutchinson, The measurement of u, th and k concentrations in building materials, Applied Radiation and Isotopes 51 (5) (1999) 483–492.

[65] Z. Szabó, P. Völgyesi, H. Nagy, C. Szabó, Z. Kis, O. Csorba, Radioactivity of natural and artificial building materials–a comparative study, Journal of Environmental Radioactivity 118 (2013) 64–74.

[66] M. J. Willis, I. R. Stewart, A. D. Nicholson, D. E. Archer, W. R. Ray, Systematic study of variation in radiation data in cluttered environments, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 954.

[67] I. R. Stewart, A. D. Nicholson, D. E. Archer, M. J. Willis, M. W. Swinney, I. Garishvili, W. R. Ray, Understanding and quantifying the systematic effects of clutter within a radiation detection scene, Journal of Radioanalytical and Nuclear Chemistry 318 (1) (2018) 727–737. doi:10.1007/s10967-018-6159-8.
URL https://doi.org/10.1007/s10967-018-6159-8

[68] R. Livesay, C. S. Blessinger, T. F. Guzzardo, P. A. Hausladen, Rain-induced increase in background radiation detected by radiation portal monitors, Journal of Environmental Radioactivity 137 (2014) 137–141.

[69] J. M. Ghawaly, A. D. Nicholson, M. J. Willis, D. E. Archer, M. Cook, H. L. Hall, An unsupervised radiation anomaly detection algorithm using a deep neural autoencoder, The Proceedings from the INMM 60th Annual Meeting.

[70] J. M. Ghawaly, A. D. Nicholson, D. E. Archer, M. T. Cook, Characterization and evaluation of the autoencoder radiation anomaly detection (arad) model, Engineering Applications of Artificial IntelligenceIn submission.

[71] B. Jeon, Y. Lee, M. Moon, J. Kim, G. Cho, Reconstruction of compton edges in plastic gamma spectra using deep autoencoder, Sensors 20 (10) (2020) 2895.

[72] E. T. Moore, J. L. Turk, W. P. Ford, N. J. Hoteling, L. S. McLean, Transfer Learning in Automated Gamma Spectral Identification, arXiv:2003.10524 [physics]ArXiv: 2003.10524.
URL `http://arxiv.org/abs/2003.10524`

[73] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[74] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[75] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[76] DOE/NNSA - Remote Sensing Laboratory, Northern virginia array (novarray), electronic (2018).

[77] G. Daniel, F. Ceraudo, O. Limousin, D. Maier, A. Meuris, Automatic and real-time identification of radionuclides in gamma-ray spectra: A new method based on convolutional neural network trained with synthetic data set, IEEE Transactions on Nuclear Science 67 (4) (2020) 644–653.

[78] M. Kamuda, J. Zhao, K. Huff, A comparison of machine learning methods for automated gamma-ray spectroscopy, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 954 (2020) 161385.

[79] P. Olmos, J. Diaz, J. Perez, G. Garcia-Belmonte, P. Gomez, V. Rodellar, Application of neural network techniques in gamma spectroscopy, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 312 (1-2) (1992) 167–173.

[80] V. Pilato, F. Tola, J. Martinez, M. Huver, Application of neural networks to quantitative spectrometry analysis, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 422 (1-3) (1999) 423–427.

[81] R. T. Kouzes, J. H. Ely, B. D. Geelhood, R. R. Hansen, E. A. Lepel, J. E. Schweppe, E. R. Siciliano, D. J. Strom, R. A. Warner, Naturally occurring radioactive materials and medical isotopes at border crossings, in: 2003 IEEE Science Symposium, 2003, pp. 1448—-1452. `doi:10.1109/NSSMIC.2003.1351967`.

[82] Z. Szabó, P. Völgyesi, H. É. Nagy, C. Szabó, Z. Kis, O. Csorba, Radioactivity of natural and artificial building materials –a comparative study, Journal of Environmental Radioactivity 118 (2013) 64–74. `doi:https://doi.org/10.1016/j.jenvrad.2012.11.008`.
URL `http://www.sciencedirect.com/science/article/pii/S0265931X12002640`

[83] R. J. McConn Jr, C. J. Gesh, R. T. Pagh, R. A. Rucker, R. G. Williams III, Compendium of material composition data for radiation transport modeling, Tech. rep., Pacific Northwest National Laboratory, Richland, Washington (2011).

[84] J. C. Wagner, D. E. Peplow, S. W. Mosher, Fw-cadis method for global and semi-global variance reduction of monte carlo radiation transport calculations, Nuclear Science and Engineering 176 (1) (2014) 37–57.

[85] D. J. Mitchell, L. T. Harding, G. G. Thoreson, S. M. Horne, Gadras detector response function, Tech. rep., Sandia National Laboratories, Albuquerque, New Mexico (2014).

[86] D. E. Archer, M. S. Bandstra, D. S. Bolme, S. L. Cleveland, J. C. Curtis, G. G. Davidson, I. Garishvili, J. O. Johnson, A. K. Mikkilineni, M. S. L. McLean, D. E. Nicholson, A Dand Peplow, A. A. Plionis, M. J. Poska, B. J. Quiter, W. R. Ray, A. J. Rowe, I. R. Stewart, M. W. Swinney, W. M. J, Mid-project report for the modeling urban scenarios and experiments (muse) project, Tech. rep., Oak Ridge National Laboratory, Oak Ridge, Tennessee (2018).

[87] A. D. Nicholson, I. Garishvili, D. E. Peplow, D. E. Archer, W. R. Ray, M. W. Swinney, M. J. Willis, G. G. Davidson, S. L. Cleveland, B. W. Patton, D. E. Hornback, J. J. Peltz, M. S. L. McLean, A. A. Plionis, B. J. Quiter, M. S. Bandstra, Multiagency urban search experiment detector and algorithm test bed, IEEE Transactions on Nuclear Science 64 (7) (2017) 1689–1695.

[88] L. Lu, C. M. Anderson-Cook, T. Ahmed, Non-uniform space-filling designs, Journal of Quality Technology (2020) to appeardoi:10.1080/00224065.2020.1727801.

[89] A. D. Nicholson, I. Garishvili, D. E. Peplow, D. E. Archer, W. R. Ray, M. W. Swinney, M. J. Willis, G. G. Davidson, S. L. Cleveland, B. W. Patton, D. E. Hornback, J. J. Peltz, M. S. L. McLean, A. A. Plionis, B. J. Quiter, M. S. Bandstra, Multi-agency urban search experiment detector and algorithm test bed, IEEE Transactions on Nuclear Science 64 (7) (2017) 1689–1695.

[90] M. W. Swinney, D. E. Peplow, B. W. Patton, A. D. Nicholson, D. E. Archer, M. J. Willis, A methodology for determining the concentration of naturally occurring radioactive materials in an urban environment, Nuclear Technology 203 (3) (2018) 325–335.

[91] R. Trevisi, S. Risica, M. D'Alessandro, D. Paradiso, C. Nuccetelli, Natural radioactivity in building materials in the european union: A database and an estimate of radiological significance, J. Environ. Radioact. 105 (11) (2012) 11–20.

[92] K. L. Myers, C. M. Anderson-Cook, D. Archer, A. Nicholson, D. E. Peplow, B. Quiter, Modeling urban search experiments data competition: Website content, Tech. rep., Los Alamos National Laboratory, Los Alamos, New Mexico (2017).

[93] A. D. Nicholson, D. E. Peplow, J. M. Ghawaly, M. J. Willis, D. E. Archer, Generation of synthetic data for a radiation detection algorithm competition, IEEE Transactions on Nuclear Science 67 (8) (2020) 1968–1975.

[94] Detecting radiological threats in urban areas, https://doi.ccs.ornl.gov/ui/doi/74, accessed: 2020-02-25.

[95] T. U. of Chicago, FAQ: Globus Connect and Endpoints (2010-2020 (accessed September 1, 2020)).
URL https://docs.globus.org/faq/globus-connect-endpoints/

[96] L. Lu, C. M. Anderson-Cook, Incorporating uncertainty for enhanced leaderboard scoring and ranking in data competitions, Quality Engineering (2020) to appear.

[97] L. Lu, C. M. Anderson-Cook, M. Zhang, Understanding the Merits of Winning Solutions from a Data Competition for Varied Sets of Objectives, Tech. rep., Los Alamos National Laboratory LAUR-20-24756 (2020).

[98] L. Lu, C. M. Anderson-Cook, T. J. Robinson, Optimization of designed experiments based on multiple criteria utilizing a pareto frontier, Technometrics 53 (4) (2011) 353–365. arXiv:https://doi.org/10.1198/TECH.2011.10087, doi:10.1198/TECH.2011.10087.
URL https://doi.org/10.1198/TECH.2011.10087

[99] C. M. Anderson-Cook, L. Lu, Weighing your options, Quality Progress 45 (10) (2012) 50–52.

[100] J. L. Chapman, L. Lu, C. M. Anderson-Cook, Process optimization for multiple responses utilizing the pareto front approach, Quality Engineering 26 (3) (2014) 253–268. arXiv:https://doi.org/10.1080/08982112.2013.852681, doi:10.1080/08982112.2013.852681.
URL https://doi.org/10.1080/08982112.2013.852681

[101] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation (2015). arXiv:1505.04597.

[102] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[103] Tin Kam Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, 1995, pp. 278–282 vol.1.

[104] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 3146–3154.
URL http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[105] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, J. Feng, Dual path networks (2017). arXiv:1707.01629.

[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, CoRR abs/1706.03762. arXiv:1706.03762.
URL http://arxiv.org/abs/1706.03762

[107] Scikit learn kmeans.

[108] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps (2013). arXiv:1312.6034.

[109] D. Misra, Mish: A self regularized non-monotonic neural activation function (2019). arXiv:1908.08681.

[110] A. M. Badshah, J. Ahmad, N. Rahim, S. W. Baik, Speech emotion recognition from spectrograms with deep convolutional neural network, in: 2017 International Conference on Platform Technology and Service (PlatCon), 2017, pp. 1–5.

[111] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.
URL https://doi.org/10.1162/neco.1997.9.8.1735

[112] J. H. Friedman, Greedy function approximation: A gradient boosting machine., Ann. Statist. 29 (5) (2001) 1189–1232. doi:10.1214/aos/1013203451.
URL https://doi.org/10.1214/aos/1013203451

# Appendices

## A 1st Place Writeup: pfr

# Detecting Radiological Threats in Urban Areas

### Algorithm description

pfr

### Introduction

This algorithm is centered around a temporal convolutional neural network operating on binned data. This network contains multiple identical subnetworks running on different temporal scales. Multiple techniques encourage generalization of the model, including data augmentation by event subsampling and weight sharing across temporal scales.

The neural network itself does not require prior knowledge of the energy spectrum of each source, and is able to handle shielded sources or mixed sources without any special handling.

## 1 Preprocessing

### 1.1 Energy binning

Energy binning was performed according to a manually constructed scheme. The goal was to minimize the number of bins in order to reduce unnecessary degrees of freedom for the learning algorithm, while maintaining separation of energy peaks from both natural and non-natural sources.

It was apparent that the simulation was performed at discrete 2 keV energy steps, so a first step bins the simulation results according to this grid, rounding any ambiguous values (namely, odd integer energies such as 13.0 keV) alternatively up or down.

Next a number of thresholds were selected so that energy levels above each successive threshold have the binning step doubled, that is, their energy resolution halved:

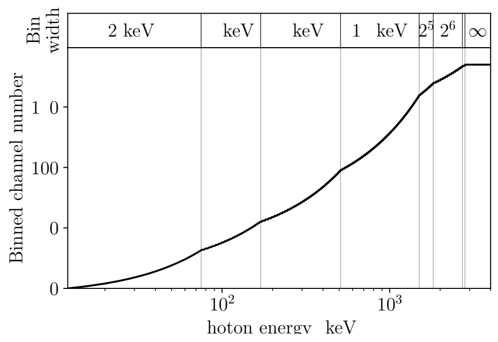


Figure 1: Energy binning scheme

The resulting scheme is approximately logarithmic, with 186 bins defining a slope on the order of 100 bins per decade of energy. The slope decreases rapidly beyond 1.8 MeV due to decreased signal level in that range. The last bin covers all energy levels above 2.811 MeV. This was entirely selected a priori based on the shapes of the spectrum of natural and non-natural sources after broadening by the NaI(Tl) detector, and was not refined based on neural network performance.

It is likely that a simpler exactly logarithmic scheme would perform well.

### 1.2 Temporal binning

The algorithm operates at three distinct scales, binning the temporal span of the entire run into 160, 80, or 40 steps. This length was chosen so that the characteristic duration of the threat signal would be on the order of 4 steps in at least one of these scales.

Normalizing the scale with respect to the entire run length has the benefit of reducing the range of temporal scales that need to be examined. In a real-world scenario with no definite run length, the equivalent would be to normalize with respect to vehicle speed, effectively turning temporal steps into spatial steps, but this wasn't an option in this contest as vehicle speed was not provided.

This scale normalization results in large variations in average signal level, as for example short run times or high vehicle speeds result in decreased photon count per cell.

### 1.3 Data augmentation

Once energy and temporal binning schemes are defined, for each run we simply count the number $n_{ij}$ of events present for each time step $i$ and energy bin $j$. This array will be the input to the neural network.

In order to enforce robustness to the Poisson statistics of $n_{ij}$, training data is augmented by randomly subsetting the event data from a given run with a probability $p$ drawn from the set $\{25\%, 60\%, 100\%\}$. Equivalently, the true cell count $n_{ij}$ is replaced with a random cell count $n'_{ij}$ drawn from the binomial distribution $B(n_{ij}, p)$. The implementation uses the former method internally, as accurate binomial sampling is costly and the event count is low.

This augmentation is also key in preventing overfit by acting as a strong regularization: it can be viewed as a form of input dropout. Finally, it contributes to making the algorithm robust to changes in signal strength, which is important due to the temporal binning scheme we employ.

Other augmentation schemes such as temporal scaling and reversal could also yield some limited improvements, but were not implemented here.

## 2 Neural network model

### 2.1 Label format

For each training example, a ground truth is generated:

1. If no source is present, the "no source" target is set to 1, and all other targets to 0.
2. If a source is present, one of the three scales is selected depending on the approximate duration of the signal. Then the $k$ target cells surrounding the closest-approach time for the given scale and the given source type are set to $1/k$, and all other targets to 0. Note that $k = 4$ is a constant and does not depend on exact signal duration: this is important in order for the estimated probabilities to be meaningful.

As there are 6 source types, this results in a flattened label dimension of:

$$\begin{aligned} D &= 1681 \\ &= 1 \text{ (no source)} \\ &+ 40 \times 6 \text{ (coarsest scale)} \\ &+ 80 \times 6 \text{ (medium scale)} \\ &+ 160 \times 6 \text{ (finest scale)} \end{aligned}$$

All ground truth label formats sum to 1, which allows viewing the problem as a soft-labeled classification problem on $D$ categories.

Note that due to selection of the scale based on approximate duration, generating the ground truth label requires not just the source time but also the offset-to-speed ratio as derived from `answerKey.csv`, which can be understood as the temporal radius $t_0$ of the signal in the ideal inverse-squared-distance signal decay formula $A(t) = \frac{A_0}{1+(t/t_0)^2}$. The FWHM duration of this ideal signal is $2t_0$. If on the other hand $t_0$ wasn't directly available, various methods could be employed to either provide surrogates for this quantity or reduce the amount of supervision provided to the algorithm. It is entirely possible that with careful design such methods might outperform the current algorithm, at the cost however of greater complexity and perhaps increased training time.

## 2.2 Multi-scale network architecture

The input is downsampled by summation of event counts to each of the three temporal scales. Identical single-scale subnetworks with shared weights are then run on each scale, as described in the next section. Each subnetwork's output operates in logarithmic probability space, that is, no softmax has been applied to it yet: it is then flattened to a vector and all three outputs are concatenated. Finally, the "no source" target is appended as the fixed scalar 0, which can be imposed without loss of generality as softmax is invariant to addition of a constant: in other words, the subnetworks are free to learn any bias they wish in order to adjust the overall probability of a source being present.

Once all $D$ target dimensions have been assembled, softmax is applied in order to estimate the true target.

The network is implemented in the PyTorch framework. Training is performed according to Kullback–Leibler divergence with an Adam optimizer. The learning rate is initially set at $10^{-3}$ and multiplied by 0.3 after epochs 20 and 30, and training is stopped at epoch 40. No weight decay is applied.

## 2.3 Single-scale subnetwork architecture

Each subnetwork is a straightforward temporal convolution network, consisting of:

- An "embedding" layer, which is really a pointwise convolution from the 186 energy bins to 32 channels, followed by ReLU activation.
- A sequence of 5 convolution layers with temporal kernel size 3 steps and output size 40 channels, with ReLU activation.
- A pointwise convolution layer outputting log-likelihoods for each of the 6 sources, with no activation.

No global pooling is used, which results in a limited receptive field of 11 time steps. This would allow real-time predictions with bounded latency, as the model does not need access to a large quantity of background events. Specifically, the latency corresponds to the raw measurement becoming available when the source approximately reaches the cone of angular diameter $2\arctan\frac{4}{11} = 40°$ around the vehicle's rear-view direction. The latency could naturally be reduced further with a design that optimizes for it, at the cost of reduced sensitivity.

Convolutions are padded so that the output has the same length as the input. Initialization is PyTorch's default weight initialization, which has limited impact since we use the Adam optimizer.

## 3 Ensembling

30 networks were trained on 90% subsets of the training data. The predicted probabilities were averaged after softmax to form a raw ensemble prediction that will be passed to post-processing.

If inference speed was a concern, it should be possible to use fewer networks with little loss in accuracy.

## 4 Post-processing

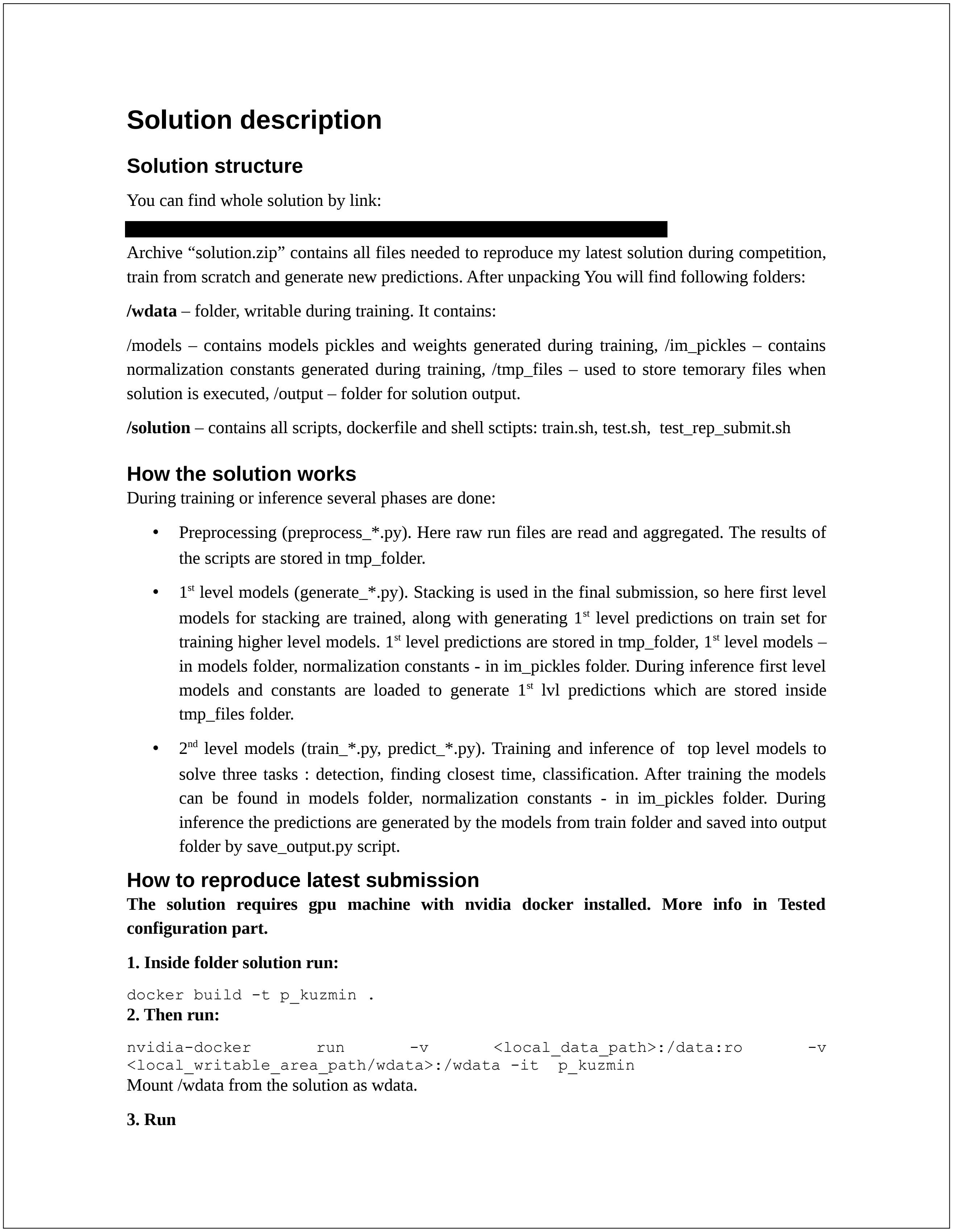
A source detection is made by the algorithm when the raw “no source” estimated probability falls under an empirically determined threshold (0.65). Note that the true probability that there is no source is in fact lower than this threshold.

If a detection is made, each subnetwork's output is expanded by nearest-neighbor upsampling so that it has as many time steps as the finest-scale network.

The source type is chosen as the one with the highest sum of expanded probabilities. Using expanded probabilities here was an oversight, as unexpanded probabilities would be more meaningful theoretically. However, this should have little noticeable effect on accuracy.

The closest-approach time is determined by first smoothing the expanded probabilities with a Gaussian kernel of standard deviation $\sigma = 7$ cells, then weighting each subnetwork's probabilities by $n^2$ where $n$ is the number of steps in its output, and finally summing across all source types to extract the time step having maximal probability. A weighting factor of $n$ is necessary to correct for the fact that the expansion step multiplies the number of cells of the coarsest subnetworks, and we increase the exponent further to $n^2$ to account for the fact that the finer-scale subnetworks have greater temporal precision, which reduces noise in the estimation of closest-approach time.

## B 2nd Place Writeup: p_kuzmin



## Solution description

### Solution structure

You can find whole solution by link:

[redacted]

Archive “solution.zip” contains all files needed to reproduce my latest solution during competition, train from scratch and generate new predictions. After unpacking You will find following folders:

**/wdata** – folder, writable during training. It contains:

/models – contains models pickles and weights generated during training, /im_pickles – contains normalization constants generated during training, /tmp_files – used to store temorary files when solution is executed, /output – folder for solution output.

**/solution** – contains all scripts, dockerfile and shell sctipts: train.sh, test.sh, test_rep_submit.sh

### How the solution works

During training or inference several phases are done:

- Preprocessing (preprocess_*.py). Here raw run files are read and aggregated. The results of the scripts are stored in tmp_folder.
- 1st level models (generate_*.py). Stacking is used in the final submission, so here first level models for stacking are trained, along with generating 1st level predictions on train set for training higher level models. 1st level predictions are stored in tmp_folder, 1st level models – in models folder, normalization constants - in im_pickles folder. During inference first level models and constants are loaded to generate 1st lvl predictions which are stored inside tmp_files folder.
- 2nd level models (train_*.py, predict_*.py). Training and inference of top level models to solve three tasks : detection, finding closest time, classification. After training the models can be found in models folder, normalization constants - in im_pickles folder. During inference the predictions are generated by the models from train folder and saved into output folder by save_output.py script.

### How to reproduce latest submission

**The solution requires gpu machine with nvidia docker installed. More info in Tested configuration part.**

**1. Inside folder solution run:**

```
docker build -t p_kuzmin .
```

**2. Then run:**

```
nvidia-docker run -v <local_data_path>:/data:ro -v <local_writable_area_path/wdata>:/wdata -it  p_kuzmin
```

Mount /wdata from the solution as wdata.

**3. Run**

```
./test_rep_submit.sh /data/testing/ <output_filename>.csv
```

**One should run script `test_rep_submit.sh` to get exactly same results as in latest solution during competition phase. Running script test.sh will lead to incorrect results. The explanation is following:**

test_rep_submit.sh script is same as test.sh except the following steps. It does not clean the tmp_files folder and does not generate predictions from three first level models. I have already put the correct files that should be generated by these first level models into the tmp_files folder. The reason for this is following: During competition phase I have trained a number of 1st level models: three boosting models and three multilayer perceptron (mlp) models. Keras package was used to train mlp models. During training, I have been saving the best model weights via Keras callback "ModelCheckpoint". Usually after training this way, one should load best weights and generate predictions with saved model. Unfortunately I've accidentally missed line of code loading the weights after training the model. So the 1st level mlp predictions were generated with the model which had weights of last training epoch. Unfortunately these weights were not saved. During competition phase I've generated these predictions for training and testing sets once and used them for higher level models without retraining first level models, so I have noticed the lack of weights, corresponding to the 1st lvl mlp predictions only when started to prepare scripts and writeup. That is why these precalculated 1st level test set predictions are needed to generate absolutely same results as my latest submission. After checking reproducibility training and testing can be done in usual way.

## Training from scratch and testing

If You wish to train all models from scratch You can run command:

```
./train.sh /data/training/ /data/trainingAnswers.csv
```

This script will erase all files inside /wdata folders and will start training and saving all the models (including three first level mlp models).

After training is done test.sh can be used to generate predictions for testing data or previously unseen data. To do this run:

```
./test.sh /data/testing/<output_filename>.csv
```

This script will erase all files in /wdata/tmp_files and /wdata/output folders and will start predicting. The result will be available in /wdata/output/<output_filename>.csv

**Additional info:**

Each of three provided scripts will output progress information to the terminal very often. It just shows that something is happening and the process is not stuck.

test.sh (after training from scratch) and test_rep_submit.sh give a bit different answers for provided test data : approximately 1% for detection task.

## Tested configuration

The solution was developed and tested on the configuration: AMD FX-8320 8-core processor, 24Gb RAM , nvidia geforce 1080Ti 11 Gb, 7200 rpm HDD, ubuntu-16.04 LTS. Training process takes about 4.17 hours. Test predicting: 1.35 hours. The solution was also tested on **p3.2xlarge** AWS instance with Deep Learning AMI (Ubuntu) Version 22.0 - ami-01a4e5be5f289dd12 .

# Models description

## Feature extraction

Each Run file was aggregated to 1 Hz rate. Two kinds of features were produced during aggregation:

- Aggregation to energy channels (channel features). The energy was binned to the energy channels. The bandwidth is set to be 50 kEv (661 kEv*0.075 = 49.575 kEv). Only first 64 channels were taken (0-3.2 MEv). Higher energies are rare in data. Then calculated the number of events in each energy channel. So each run can be presented as a 64 channel signal with 1s sampling period.
- A number of statistics (statistics features) for time between events and event energy inside aggregation window (1s): mean,max,min,median,std,skew and also count of events in a window. These features are extracted by preprocess_*.py files. Later only part of these features is used and some additional features are created in several files (for example in generate_lgb_features_stats.py function fe()): events count, energy mean, energy median, energy std, energy skewness, difference between energy mean and median,something like signal-noise ratio: the ratio of energy mean and energy std, and maximal signal-noise ratio.

## Validation scheme

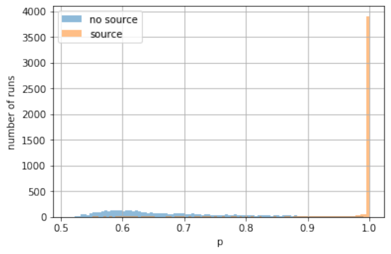
5 fold cross validation (CV) with stratification by source type was used during training and validation. The models trained in each split of CV are saved. Test predictions are done by averaging the results of such one type models. The random seed for CV was fixed and used the same each time.

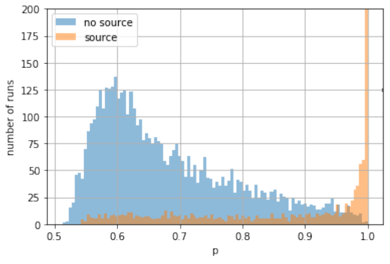
## First level models

Six first level models are trained in the following way.

One can assume that each second of each run is an absolutely independent observation (forget that we have a number of time-series). Then each observation is labeled as source type of run id it took from (1-6) or for binary classification 1/0 (source/no source) .

After that it is possible to train a classifier on a huge amount of points. While training auc metric was used as a score for binary classifiers and accuracy for multiclass. The score of these models is very low (~0.55 auc for example). But if we check the maximum of the classifier outputs p for each run we'll find out that a huge part of runs with sources has one or more points where classifier is very confident that there is a source, while maximum confidence for no source is lower:





Even simple one model baseline can be done here. Choosing an appropriate threshold for p (about 0.97) can give about 0.93 accuracy for detection task on CV for train data. We can also mark the point with highest p as the closest to the source point.

So 4 models are trained to predict whether single point is from run with source or without:

- gradiend boosting model in light gbm package (lgb) on  channel features
- lgb on  statistics features
- multilayer perceptron (mlp) in keras package on  channel features
- mlp on  statistics features

Mlp models were trained with decreasing learning rate (like in cosine annealing) for 10 epochs  for each CV split.

Also two models for multi class classification were trained.

- lgb on  channel features
- mlp on  channel features

The runs with source only were used for training multiclass classifiers. Again simple baseline possible: just classifying each point of the run and then assign the class of the source to the most frequent class predicted by the model. The accuracy of such baseline is about 0.87 for lgb model.

## Second level models

Three different models were trained for the tasks. All of them are trained on channel features combined with the 1st level predictions. All the models were trained in the 5 fold CV.

### Detection

All the signals were padded with zeros to the size of 1024. The number of channel features was lowered to 16 (by summing counts in neighbor channels) before combining with the 1st level predicts. All the data was scaled with min-max scaling. Then continuous wavelet transform was used for each channel with “mexican hat” mother wavelet to get 30 channels from each channel. To decrease the amount of data the maximum from these each 30 channels in each moment was used to recreate again 1 channel. (transformation: number of channels → 30*number of channels (each channel transformed) → number of channels (maximum  in every 30 channels is taken))

The detection model is 1D convolution network with attention layer and one fully connected layer after attention:

Input_layer, Conv1D, Conv1D, MaxPooling, Dropout, Conv1D, Conv1D, MaxPooling, Conv1D, Conv1D,Attention,Dense, Dropout, output_layer

The model was trained with binary focal loss. Focal loss is used to pay more attention to difficult for classification cases. While training snapshot ensembling was also used. Total number of epochs: 50, periods: 5. Cosine annealing for changing learning rate was used. So after CV 25 (5 splits*5 periods) models are presented. The predictions of models for a run are averaged and I used threshold 0.5 to classify if run with/without source. Even though we know that competition metric penalizes more for FP finding threshold for detection can be not a good idea. We have great imbalance of hard cases in train and test data. Leaderboard probing is possible but also can lead to the overfeat. So I just tried to build as accurate classifier as possible on available data.

## Finding closest point to the source

The training was done only on data with sources. Channel features were combined with 1st level predictions and min-max scaled. While training cropping (with probability 0.5) random size fragment with closest point to the source was used. Then each channel was standardized inside such run and transformed to the length of 1024 with interpolation. Such multichannel signals (cropped or not) then labeled with a 1-0 mask: a random region near the closest point to the target is labeled with 1.

The model has 1d U-net structure with binary focal loss. While training decreasing the learning rate on plateau is used and best model is saved. Each epoch the score was calculated in the following way.

Getting predicted closest time:

- get prediction mask for each sample a 1024 array for one signal.
- Assign 0 to all points with p (model score) less than 0.8*maximal p
- Calculate closest point: $t_{pred} = \frac{\sum t_i p_i}{\sum p_i}$ $t$ – means time 0 to 1024
- Recalculate to the seconds (remember we scaled our signal to 1024 points).
- Clip the time by 30 seconds.

Then score is calculated: -2 points for high difference to the ground truth ( I used max difference for this 4 seconds), +0...1 points by cosine law (1 in the center, zero at +- 4) for predicted time close to the ground truth. I have also divided this score by the number of points to get normalized score.

5 models for 5 CV splits were trained. During inference the predictions of these models (masks with p) are averaged and closest time is predicted according to the procedure like in training, but before clipping to 30 seconds all points that have predicted source time less then 26 are marked as no source points.

## Classification

For classification channel features were combined with 1st level predictions on channel features and aggregated by each run with different aggregation functions: mean, median, max, min, std,skewness. The training is done only on data with source presented. Trained model – gradient boosting over decision trees from light gbm package.

For inference the predictions of 5 models are averaged and argmax is used (usual multi class classification with softmax).

So final pipline: preprocessing→ generating 1st level predictions → detecting source→ calculating closest time → for non zero detected source classify it.

It’s worth noting that simple baseline with two lgb 1st level models can give the score of about 80-83 on public leader board without any heavy training

## Possible improvements and further search

- 1st level predictions of mlp can be trained better: save best epoch/train more with cosine annealing/snapshot ensembling.
- All wavelet channels can be used to form 2D representation of the signals to train 2D conv nets on.
- Window Fourier Transform possible can help to generate a little bit different set of features.
- Maybe searching better aggregation window during preprocessing can also help.
- Meta features on the base of Fourier transform can be used and fed into the additional input of the networks (for detection and classification)
- The code needs refactoring. Preprocessing can be done twice faster minimum (just process all features in one loop)

# C 3rd Place Writeup: 03-gardn999

## Urban Nuclear Detection Challenge

[redacted] (gardn999)

**Introduction:**

The purpose of this challenge is to accurately locate various types of radiation sources. Monte Carlo particle transport models are used to simulate the presence of these sources along a city street. Each data file contains a single run with a detector moving along the street at constant speed. The gamma ray detection events are recorded as an energy in KeV and time since the last recording in microseconds. There are 6 source types simulated with the sixth being a combination of the first and fifth. Identification is made more difficult due to the presence of varied levels of background radiation from other sources and the possibility of shielding from obstacles along the street. For training, 9700 runs are generated with source type and location information provided. 4900 of these runs contain no source at all, while the remaining 4800 simulate a single source with 800 for each type. For testing, 15,840 runs with no source information provided are used.

**Solution Overview:**

This solution is written in Java with five main files. These are Main.java, Testing.java, Run.java, RunPeriod.java and RFRegressor.java. Main.java includes the methods needed for training and making predictions in the final version. All methods not necessary for the final version were moved to Testing.java. These include methods needed for solution development and making the plots at the end. A Run object processes a data file containing a single run. It uses the energy and time delay data to produce a list of RunPeriods, one for each half second interval in a run. A RunPeriod keeps track of counts vs energy for that time period. RFRegressor is a general purpose regression based random forest predictor using RunPeriod produced features for training.

The first part of the solution involves using the training data and truth information to produce the random forest model file: 'rfModel.gz'. Features are defined for each RunPeriod of every run in the training data set and input, along with truth information, into the random forest algorithm. The resulting trained random forest model is used to produce a series of probability predictions as a function of time period.

For the second part, predictions for source type and location are made using these probabilities vs. time period. Ideally, a radiation source will produce a nice bell curve like shape centered at the correct position as shown in plots 5 and 6. If the probability magnitudes are all too small, then no source is predicted. Otherwise, the source type is identified as the corresponding distribution with the largest probability sum. A probability weighted mean for the identified source type is used to estimate a time location.

**Training:**

The 60 counts vs energy bins in RunPeriod are the primary features used in training. The characteristic energies from the different radiation source decays are noticeable in the plots. The first 4 plots show these elevated counts for source types 1 to 4. The energy bins are defined as:

counts vs energy bin = sqrt(energy(in KeV) * 2) rounded down to the nearest integer.

This means that bins for high energies cover a larger energy range. This helps to provide an adequate number of counts at higher energies where the counts vs energy rate is much lower.

For use as features, adding a quarter of the counts from the previous and next RunPeriods was found to improve the training result. Average energy and maximum energy for the period are also included for a total of 62 features.

A total of 6 random forests are trained, one for each source type. The goal of each random forest is to predict whether the corresponding source type is present in the run and, if so, which time periods are close to it. This requires defining truth information for each source type for each RunPeriod in each run. If a source type is not in the run or not near a RunPeriod then its truth value is 0. The nearness to a RunPeriod is defined as:

dt = abs(true time location – time at the center of the RunPeriod)

A RunPeriod has a larger than 0 truth value if dt is less than 10 seconds. Truth value is defined as:

truth = ( 1- dt / (10 seconds) ) ^ 2

which is a parabolic shape on each side with a sharp peak and rapid drop off. The 62 features and truth values defined for each RunPeriod for all runs in the training data are used to train the random forests. The output of this is the model file: 'rfModel.gz'.

**Making Predictions:**

After training is finished, the resulting model file is used to obtain a probability distribution for each source type for each run. These distributions are used to predict the source type and time location. Examples of these probability vs RunPeriod distributions can be seen in plots 5 to 12 at the end. Plots 5 and 6 show close to ideal cases with a roughly bell curve like shape and the true and predicted times closely matched at the center.

For determining the source type, the maximum sum of any 7 adjacent bins is found for each of the source distributions. The source type with the corresponding distribution with largest maximum sum is selected. That is, unless all maximums are too small, in which case the source type is set to 0. The greatest type identification error occurs in distinguishing between source types 1, 5 and 6. This isn't a surprise since source type 6 is a combination of 1 and 5. Plots 7 and 8 show source type 6 being misidentified as type 1. It can be seen that plot 7 for source 1 and plot 8 for source 6 are very close to the same overall magnitude. Plot 9 shows a false positive for source 2. Like most false positives, the magnitude is very small and could be eliminated with a slightly tighter cut.

Once the source type is determined, the corresponding probability distribution is used to determine source time. For this, probability is raised to the power of 7. This puts much more emphasis on the highest probability bins. The 80 adjacent bins with the largest prob^7 sum are selected. The prob^7 and (prob^7 * time) sums are used to obtain a weighted average time:

predicted time = (sum (prob^7 * time)) / (sum prob^7).

Plots 10 and 11 demonstrate that this estimate can be resilient to quirks in the distribution and plot 10 especially demonstrates why such a large number of bins are helpful.

**Deployment:**

This solution can be built with Docker using the following commands while in the solution directory:

```
docker build -t <id> .     (be sure to include the period)
```

And to run using Docker:

```
docker run -v <local_data_path>:/data:ro -v <local_writeable_area_path>:/wdata -it <id>
```

This installs Java 8 in an ubuntu shell where the solution can be run using the scripts:

for training:

```
train.sh <data_folder> <ground_truth_file>
```

for testing:

```
test.sh <data_folder> <output_file>
```

The first row of the output_file is the header: RunID, SourceID, SourceTime. The rest of the rows contain source type and time predictions for each run in the <data_folder> sorted numerically by runID.

**Performance and Requirements:**

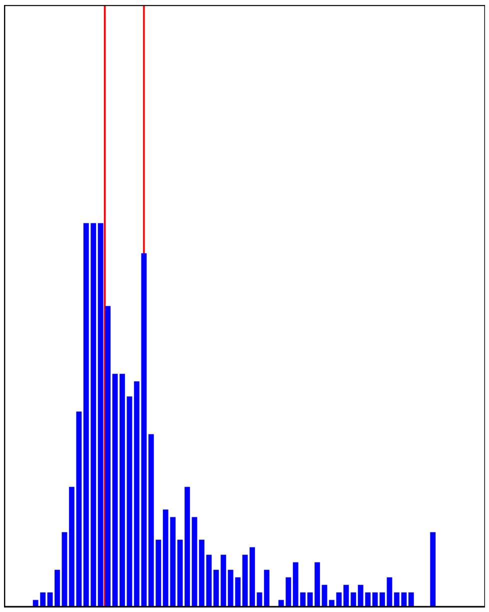
Training requires roughly 45 minutes and 4 GB of ram. Testing requires roughly 20 minutes for the 15,840 events in the testing data set and 1 GB of ram. Disk requirements are under 100 MB.

**Potential Improvements:**

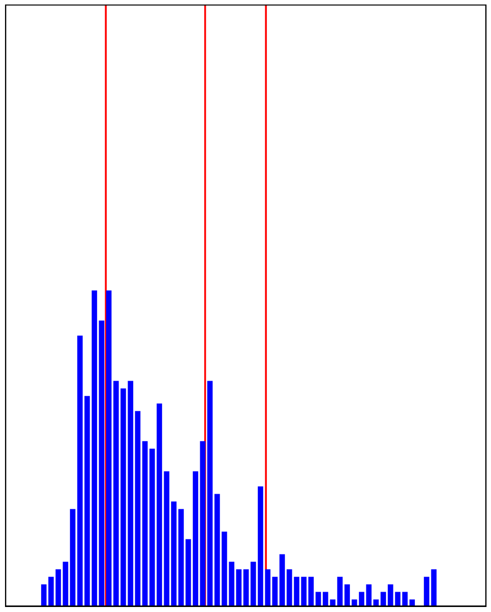
Plot 12 is an example of a run in which the source time could have been more accurately located by further analysis of the probability distributions. There are many cases such as this in which the source appears to be blocked on one side. This causes the measured time to shift away from the obstruction. In other cases the obstruction is directly in front of the source causing two peaks as seen in plots 10 and 11.

Training on more challenging runs similar to those present in the testing data set should improve the result.

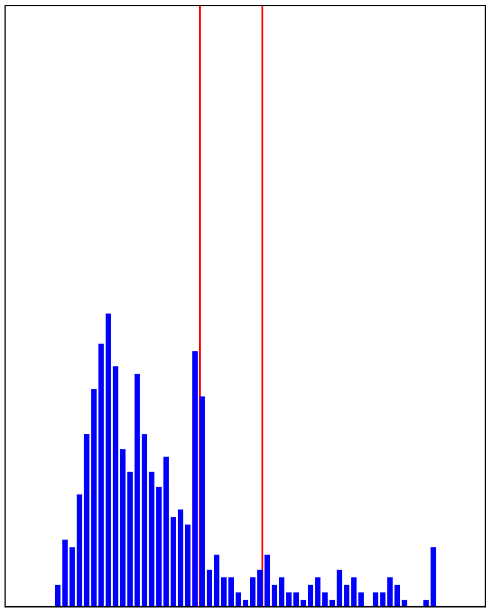
It should also be possible to better distinguish source type 6 from 1 and 5.



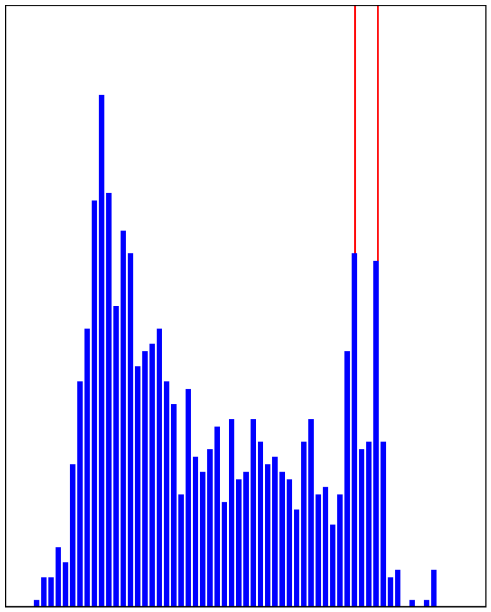
**Plot 1:** Counts vs Energy for Source Type 1
(red lines are expected peaks)



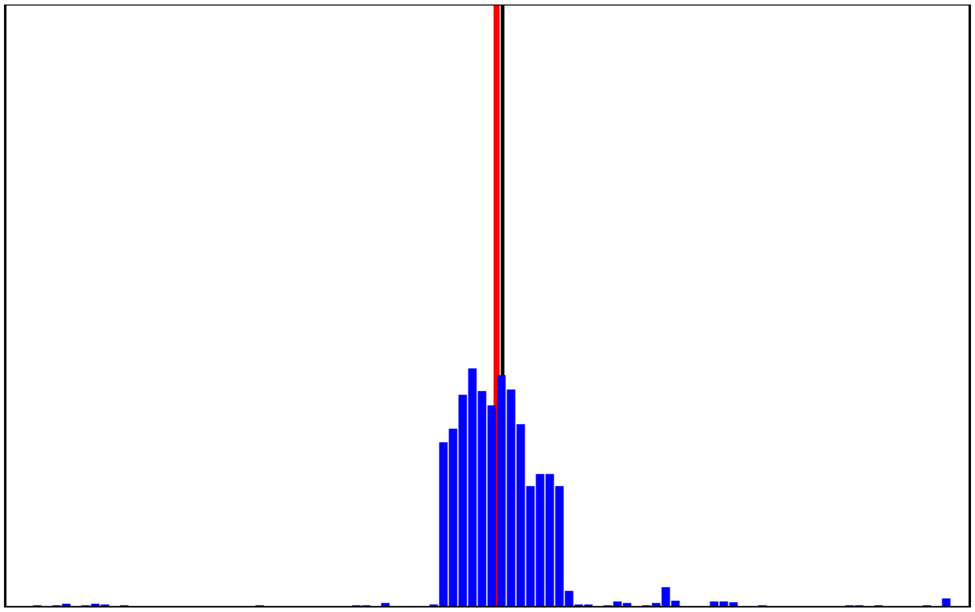
**Plot 2:** Counts vs Energy for Source Type 2
(red lines are expected peaks)



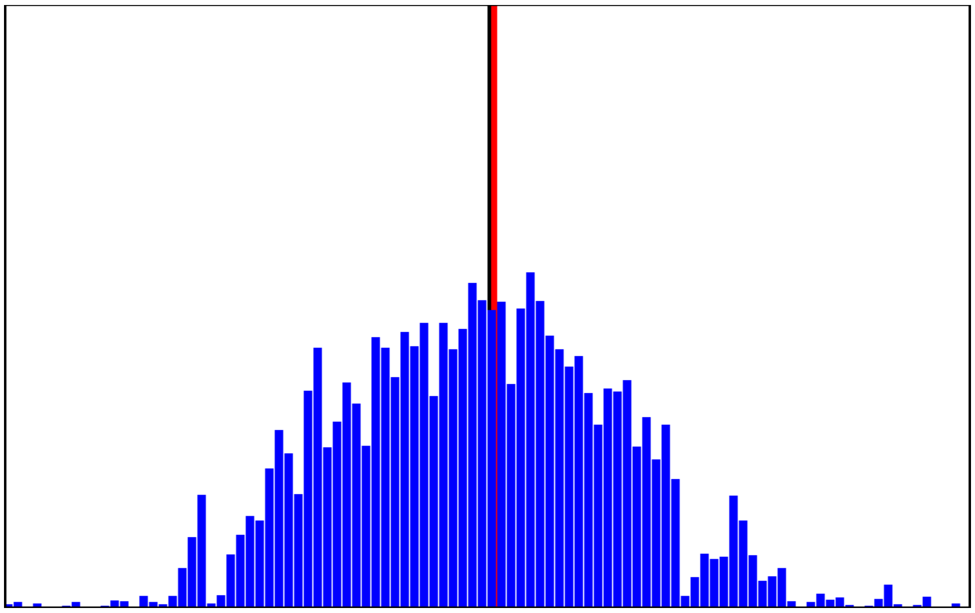
**Plot 3:** Counts vs Energy for Source Type 3
(red lines are expected peaks)



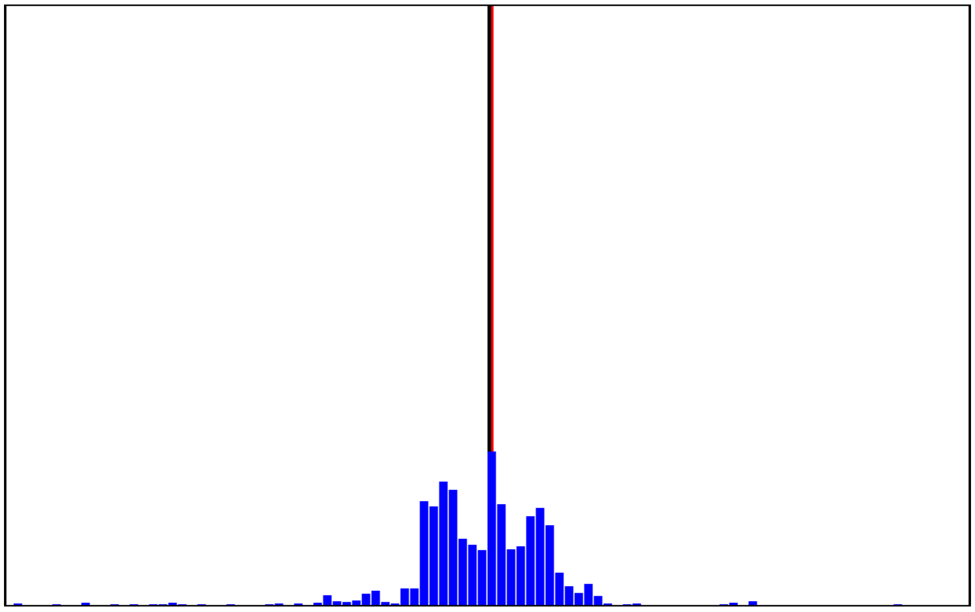
**Plot 4:** Counts vs Energy for Source Type 4
(red lines are expected peaks)



**Plot 5**: Probability vs Time Period (red line = prediction, black line = true time)

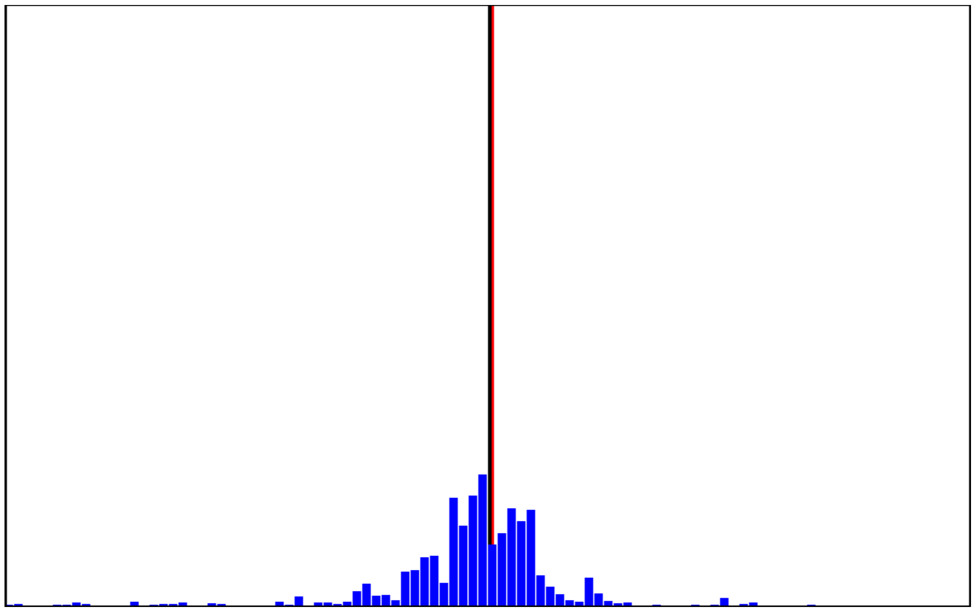


**Plot 6**: Probability vs Time Period (red line = prediction, black line = true time)



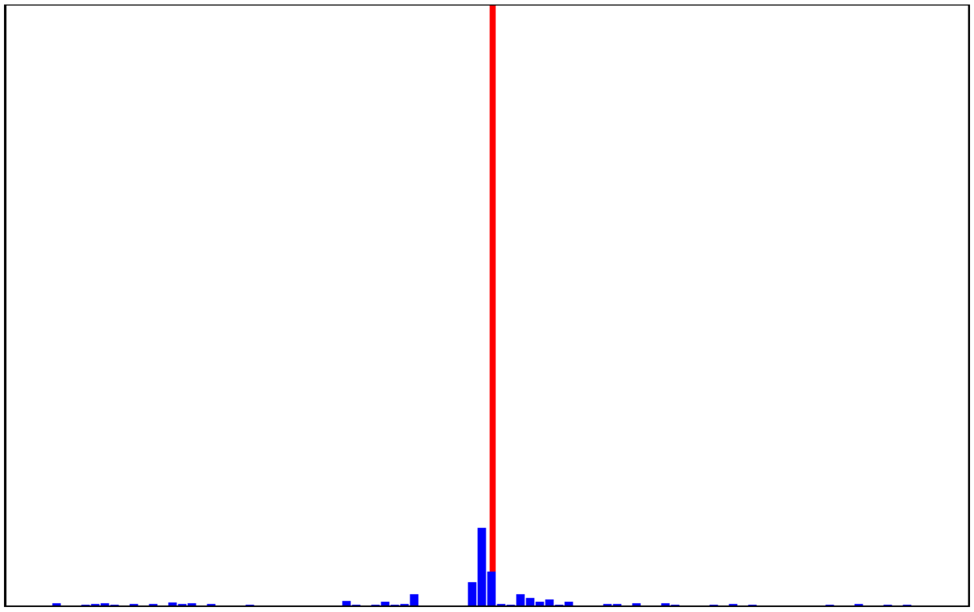
**Plot 7**: Probability vs Time Period (red line = prediction, black line = true time)

misided as Source Type 1 (should be Type 6)



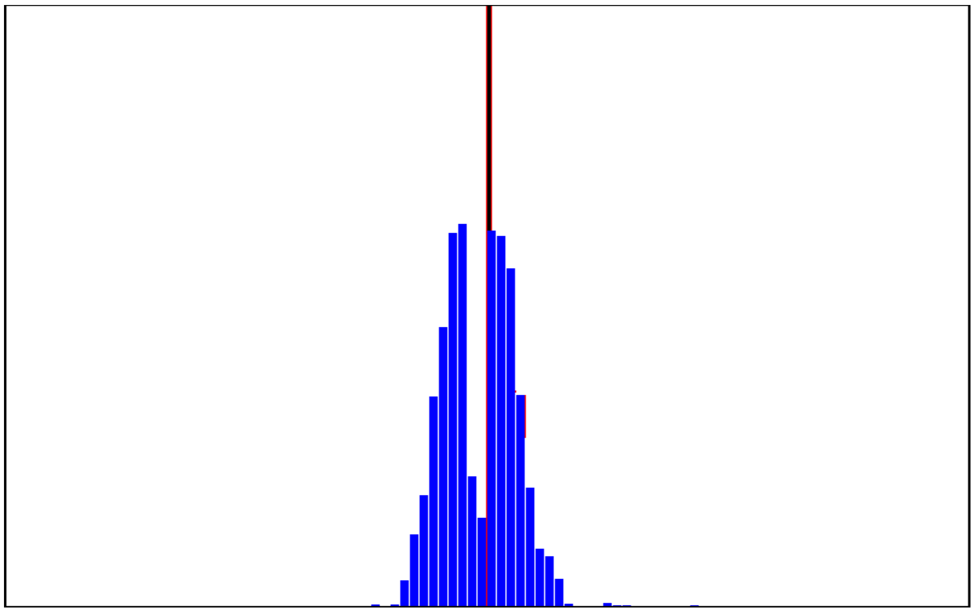
**Plot 8**: Probability vs Time Period (red line = prediction, black line = true time)

the correct distribution (Source Type 6) with slightly smaller overall magnitude



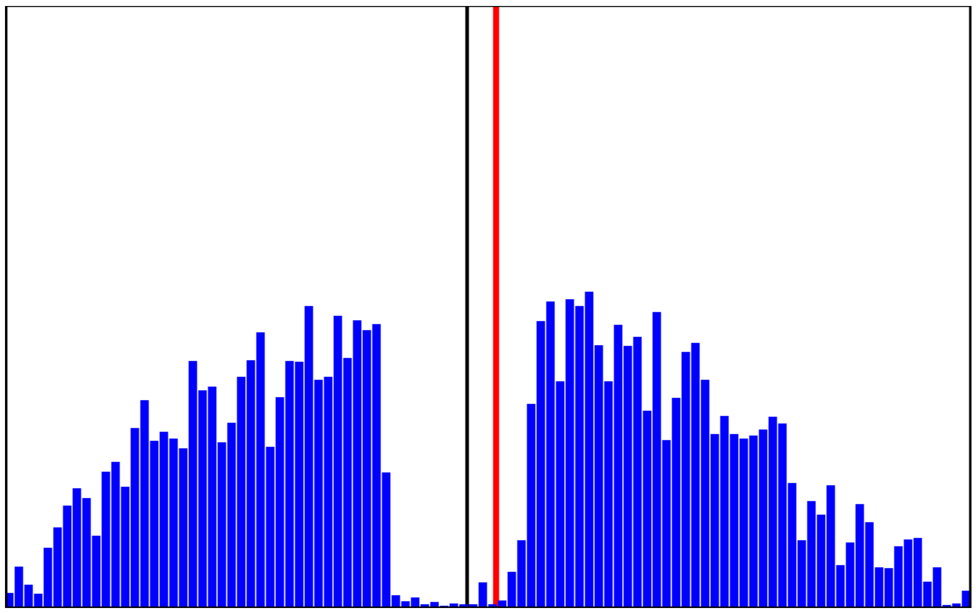
**Plot 9**: Probability vs Time Period (red line = prediction, black line = true time)

false positive (misided as Source Type 2)



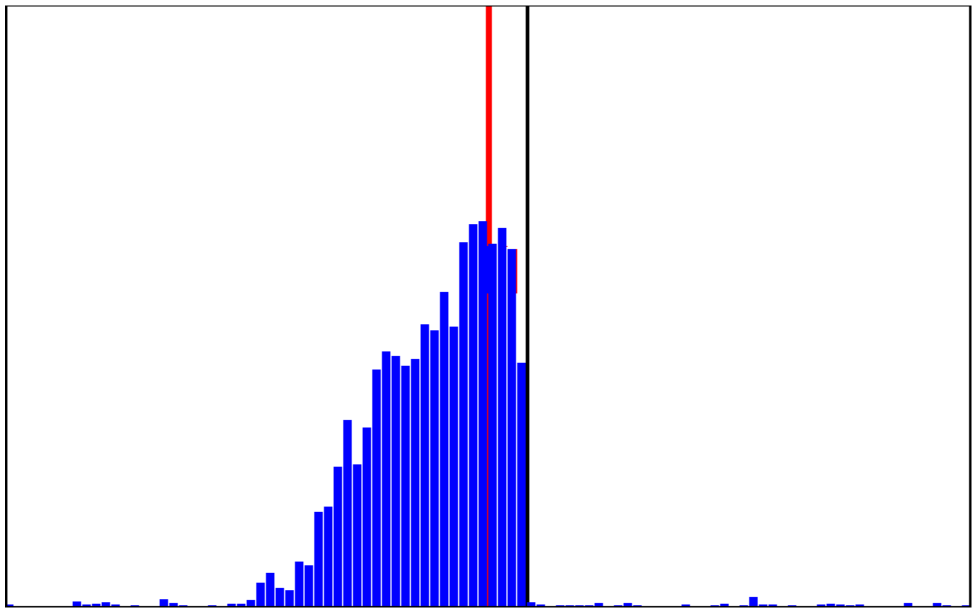
**Plot 10**: Probability vs Time Period (red line = prediction, black line = true time)

good prediction despite small gap in the center



**Plot 11**: Probability vs Time Period (red line = prediction, black line = true time)

reasonably good prediction considering size of gap



**Plot 12**: Probability vs Time Period (red line = prediction, black line = true time)

demonstrates room for improvement in cases with shielding on one side

## D 4th Place Writeup: rayvanve

# Detecting Radiological Threats in Urban Areas

rayvanve — [redacted]

May 1, 2019

## 1 Introduction

In the data challenge *Detecting Radiological Threats in Urban Areas*, one has to detect and identify illicit nuclear materials. A gamma-ray spectrometer is used to detect radioactive energy, measured in keV, which I also denote by $K$. By equipping a car with such a spectrometer, one can gather measurements from which it should be possible to identify whether there is a threat; what type of threat there is; and what the location of the threat is.

The given training dataset contains 9700 example drives for which the presence and location of a nuclear threat is given. An example (drive) consists of a datafile with timestamps and the energy detected at that moment.

My solution is based on the following pipeline:

- Data representation. By binning the data with respect to time and keV, i.e. $t$ and $K$, it is possible to view the data as an matrix or *image*. I apply a Gaussian filter to reduce the input noise;
- Feature extraction:
  - Heuristic feature extraction. Given the energy spectra of the sought sources, I calculate the similarity between the detected spectra and the source spectra using various measures;
  - CNN feature extraction. I trained *Convolutional Neural Networks* (CNNs) to predict the source (types), using a sliding window approach.
- Final prediction. After generating these features for a bunch of different parameters, I used a random forest classifier to predict the source (type), and used a simple thresholding heuristic for location detection.

## 2 Data representation

Ideally, you want to measure the exact radioactive energy at each time step, allowing you to compare the measured energy to that of a source, e.g. Figure 2.

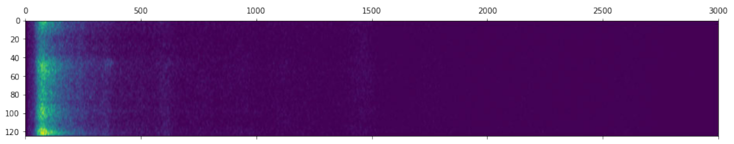


Figure 1: Image representation of run 107230 using Gaussian parameters $\sigma_t = 1.0$ and $\sigma_K = 2$. This run has a threat of type *131I* around time 47.5, which is not obvious from this graph.

Since the gamma-ray spectrometer has only discrete sampling resolution, this is impossible.

By 'binning' you can view the data as a matrix or image, and approximate such a energy spectrum. More concretely, for each run I calculated a matrix $M$, where $M_{tK}$ is the number of received counts in some time interval $[t, t + \Delta t]$ and energy interval $[K, K + \Delta K]$, for some small $\Delta t$ and $\Delta K$.

From what I understood, the gamma-spectrometer has detection variances given by a normal distribution. Therefore — and to reduce noise — I filtered the matrix $M$ by convolving it with a Gaussian kernel, parametrized with standard deviations $\sigma_t$ and $\sigma_K$, for the time- resp. energy axis. After convolving the matrix, I downsampled the matrix for practical reasons — there is no longer a need for a very fine matrix, since the information is smoothened. An example of the resulting matrix is given in Figure 1.

By comparing the source spectra to a received data spectrum (a row in the matrix $M$), one hopes to find similarities or dissimilarities. Figure 2 displays such a received data spectrum, after binning and filtering the input. It's clearly hard to answer the main questions by simply looking at these graphs.

Moreover, a priori it is not clear what parameters you should choose for $\sigma_t$ and $\sigma_K$. Playing around with these showed that $\sigma_K$ is pretty insignificant, whereas $\sigma_t$ has a big influence on the feature generation (described in the next section).

## 3 Feature extraction

I wasn't able to figure out a smart way to create an end-to-end Machine Learning approach. The main problem being that the examples have varying time lengths. I tackled this by generating features for each timestep, and then use simple operators, e.g. *max* or *min*, to turn these lists into single values.

More specifically, I came up with two different kind of features.

### 3.1 Heuristics

The various source (threat) spectra can be represented as a vector $\vec{S}_{x,y}$, where $x$ is the source type $x \in \{\text{HEU, WGPu, 131I, 60Co, 99mTc}\}$, and $y \in \{0, 1\}$

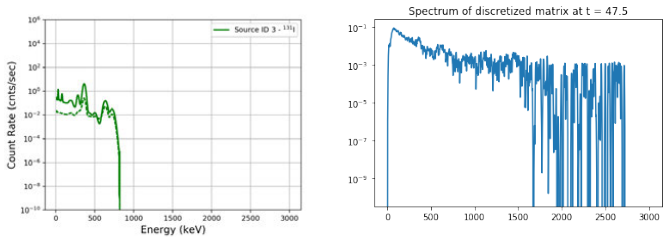


Figure 2: On the left: the energy spectra for the third threat, 131I — iodine. On the right: the energy spectrum of the transformed data at time step $t = 47.5$, the exact time step where the source should be located.

indicates whether it's shielded.

Similarly, let the data be represented as a matrix convolved with a Gaussian kernel parametrized by $(\sigma_t, \sigma_K)$. Then, for each time step $t$ the spectrum of data, a row in the matrix $M$, can be viewed as a vector $\vec{K}_t$.

#### 3.1.1 Distances

For each time step $t$ and threat $(x, y)$, I calculated a lot of metrics measuring the similarity between $\vec{K}_t$ and $\vec{S}_{x,y}$, a few that worked well were:

**Hellinger distance** I got the best results using the *Hellinger distance*. For every (discretized) time step $t$, the Hellinger distance between $\vec{K}_t$ and $\vec{S}_{x,y}$ reads as

$$\frac{1}{\sqrt{2}} \left\| \left( \frac{\vec{K}_t}{\|\vec{K}_t\|_1} \right)^{1/2} - \left( \frac{\vec{S_{x,y}}}{\|\vec{S}_{x,y}\|_1} \right)^{1/2} \right\|_2 .$$

Now, (local) minima of this function should correspond to the potential existence of a threat. Figure 3 illustrates that this measure is indicative for the source presence/type.

**Cosine similarity** Another reasonable similarity measure is the *angle* between the data and the source types, i.e.

$$\frac{\vec{K}_t \cdot \vec{S}_{x,y}}{\|\vec{K}_t\|_2 \|\vec{S}_{x,y}\|_2}.$$

Furthermore, I also calculated the cosine similarity by replacing $\|\cdot\|_2$ with the weighted $L_2$-norm

$$\|\vec{v}\|_{2,w} := \sqrt{\sum_K (\vec{w}_K)^{-1} \cdot (\vec{v}_K)^2}, \quad \vec{w} = \frac{1}{N} \sum_t \vec{K}_t,$$

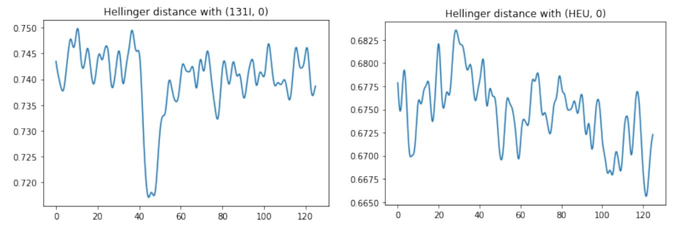


Figure 3: The calculated Hellinger distances for run 107230, preprocessed with Gaussian parameters $\sigma_t = 1.0$ and $\sigma_K = 2$. This run has a source of type 131I around time 47.5. The plots display the Hellinger distance on the $y$-axis against the time $t$ on the $x$-axis.
The left plot displays the distance of $\vec{K}_t$ to $S_{131\text{I},0}$, the energy spectrum of the unshielded 131I threat; it has a clear minimum around the correct time step. The right plot shows the distance to $S_{\text{HEU},0}$; as desired, there is no clear extrema in the graph.

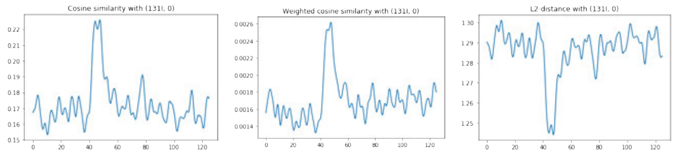


Figure 4: In the same situation as before, the images from left to right illustrate: the cosine similarity, the weighted cosine similarity and the $L_2$-distance. As before, the peaks in this graph correspond to the source location.

that is, by weighting the norm with the inverse of the mean received energy spectra. A peak in these cosine similarity measures should correspond to a source threat, see Figure 4.

**Normal distances** I also simply calculate the $L_2$-distance, i.e.

$$\left\| \frac{\vec{K}_t}{\|\vec{K}_t\|_2} - \frac{\vec{S}_{x,y}}{\|\vec{S}_{x,y}\|_2} \right\|_2 .$$

This gives reasonable results, but not as good as the ones described above.

The above measures are implemented in the `source_inner_products()` method in the file `framework.py`.

### 3.1.2 From distance sequences to features

I wanted to feed the entire distance sequences, cf. Figure 3 and 4, into a Machine Learning model in order to make final predictions. Unfortunately, I couldn't figure out a clean way to do this. Instead, I calculated some simple statistics of the sequences: *max, min, mean, std, kurtosis, argmax, argmin, snr, max* after *detrending* the signal, *min* after *detrended*, etc. These quantities capture some (important) relevant properties, while being fixed-dimensional and therefore Machine Learning friendly.

By detrending the sequence, i.e. subtracting its mean, I hoped to eliminate the background noise — this worked reasonably well.

An issue with the these distance features: the sixth source type is not available. I hoped that a machine learning model would be able to distinguish between the different types on basis of (enough) features.

#### 3.1.3 Multi scale features

The cars that collect the data have varying speed, and sources have different distances to the road. This implies that relevant features will appear at different (time) smoothing scales. From some sources we will receive radioactivity for a long period, while others are only seen for a small amount of time.

To tackle this, I generated multiple images for each example by varying $\sigma_t$, the Gaussian standard deviation in the $t$-axis. I generated the features described above for each of the different smoothing scales $\sigma_t \in \{0.15, 0.25, 0.5, 0.5, 0.75, 1.0, 1.25, 1.5, 3.0\}$.

### 3.2 CNN

The above features work reasonably well, but have two main downsides: calculating such distances requires an energy spectrum for every threat type; the features distances are calculated time stepwise, ignoring information in the surrounding time steps.

To overcome these issues, I trained a Convolutional Neural Network on the input data, because CNNs are quite well suited for processing spatial information. By training the network directly on the input data, I eliminated the need for threat energy spectra.

#### 3.2.1 Data input

I presumed that the matrix representation that I used in the previous feature generation, was not well suited for Neural Nets. Looking at Figure 1, it is clear that most of the interesting information is contained in a relatively small energy span, say keV between $20 - 500$. Moreover, the gamma-spectrometer has a lower detection resolution for the higher energies, i.e. there is a higher variance in measurements for higher energies.

I figured this could be solved by creating bins, in the energy-axis, with increasing width. By increasing the widths of keV bins linearly, one gets an image with more detail in the lower keV-region, see Figure 5. Note that these images are still smoothened by applying a Gaussian kernel. The details indeed seem to be better spread in these images.

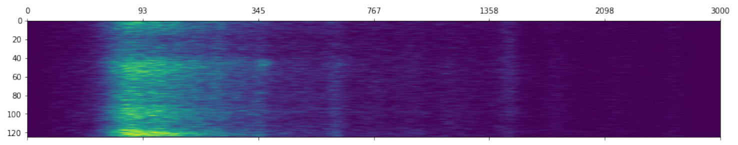


Figure 5: Another image representation of run 107230 using Gaussian parameters $\sigma_t = 1.0$ and $\sigma_K = 1$, cf. Figure 1. This image is generated using 250 bins in the keV axis, with the bin width linearly increasing.

### 3.2.2 Model setup

The varying input length of an image is an issue: (normal) Neural Nets take a fixed size input. I tried combining a CNN with a RNN for creating an end-to-end machine learning model, but that didn't give any reasonable results. Instead, I went for a sliding window approach: I train a CNN model to predict the source (type) based on a fixed size part (window) of the data; then in the prediction phase I slide this window over the entire input sequence.

A network layout that turned out to be quite successful, is one where the first convolutional layer has a kernel that spans (almost) the entire keV-axis. Intuitively, this corresponds to a model where the first layer takes various inner products with the keV-axis. This approach is very similar to the heuristic approach described above, however, now we do not require explicit knowledge of the energy spectra of the threats. In the appendix such a model is visualized; the architecture layout is generated in the method `get_model()`.

After training such a CNN model, I use its predictions as features for my final classifier. That is, by sliding a window over the input matrix, you get classification predictions for each of the time steps. Given the list of results, I use the same tricks as before to turn this into fixed-dimensional features, i.e. by calculating *max, min, mean, std, kurtosis, argmax, argmin, snr*, etc.

Again, there are a lot of hyperparameters to be chosen, with the most import ones being the number of epochs, the window size, the time smoothing standard deviation $\sigma_t$, and the number of bins in the keV-axis. Because I didn't figure out a way to let a single model incorporate all of these parameters, I simply trained a bunch of models for a reasonable combination of these hyperparameters. From all these models, I picked the 30 best performing ones, and used these models to generate features, which I used in my final model to do the actual predictions.

In terms of epochs, I started with training the CNNs using a reasonable number of epochs 8–14. However, after handing in solutions based on these models, it turned out that running the neural nets for way longer, like 40–60 epochs, gave an improved provisional score.

## 4 Final predictions

The above methods generate a bunch of features for each of the examples. Given this set of features, all we have to do is actually answer the questions:

- does this example contain a source, and if so, what is the type of source;
- if it contains a source, what is the location of the source.

I answered these questions separately.

### 4.1 Source (type) prediction

For the source (type) prediction, I simply fed all the features generated above into a random forest classifier. First, I used such a random forest to predict whether the example contains a source (threat). If so, I used another random forest to predict the actual source type.

My idea was that a random forest classifier is basically a simple thresholding algorithm, which should perform reasonable for the kind of features I've generated. An question like 'does this example have a Hellinger distance below value X?', can be used to exclude or include certain source (types). To circumvent feature noise, I included a feature elimination step that removes some of the nonsense features.

It could be possible that a different machine learning algorithm performs way better. I didn't really spend time investigating this, since most of the gains are probably gathered from generating betting features.

### 4.2 Source location prediction

I had some serious problems creating an accurate location predictor. The above model performed reasonably well in predicting the source (type), but it doesn't give any insight into where the source is located.

My final location prediction basically looks at the minimum Hellinger distances. Recall that for each example I generated a bunch of images by varying the Gaussian smoothing parameter $\sigma_t$. On the training set, I calculated thresholds such that a Hellinger distance below the thresholds (always) corresponds to a source. By comparing the actual Hellinger distances to these thresholds, I (more or less) return the location of the most indicative Hellinger distance. This is implemented in `predict_location_thresholds_new`.

I also tried predicting the location using CNNs, by augmenting the network with an extra output node that corresponds to the location of the source inside the window. Although this gave reasonable results, the Hellinger heuristic turned out to be a better location predictor. I ended up using the CNNs location prediction only if the above heuristic was based on a very 'weak' signal.

## 5 Conclusion

All in all, my approach gave reasonable results. I'm not really charmed by the approach, because it's based on a lot of different ingredients. It would've been way cleaner to train an end-to-end model, but I couldn't come up with a way to do this — my RNNs failed to give any sensible results. I was quite surprised that simply using some heuristics, e.g. the Hellinger distance and the cosine similarity, in combination with a simple random forest classifier gave a reasonable provisional score. The CNNs boosted my provisional score only by a few points.

It was a fun challenge to work on! The problem is very difficult allowing for a lot of different solution methods.

### 5.1 Flaws

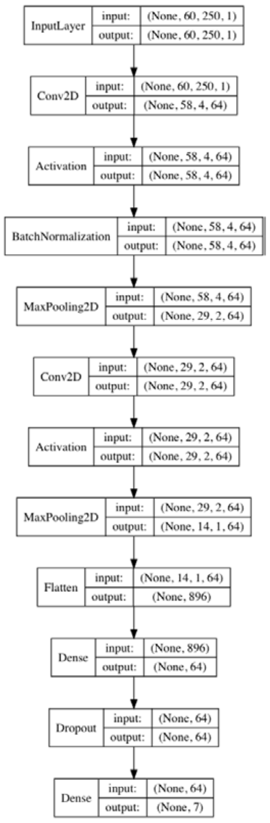
I was too lazy to properly setup a train, test and validation data split. This prevented me to do proper generalization analysis. Counter-intuitively, by handing in solutions based on CNN models that were trained for 40+ epochs, my provisional score improved. I suspect that these models sort of generalized for the provisional dataset, but failed to generalize for the final dataset.

I wonder whether some of my submissions with lower provisional scores, actually higher final scores.

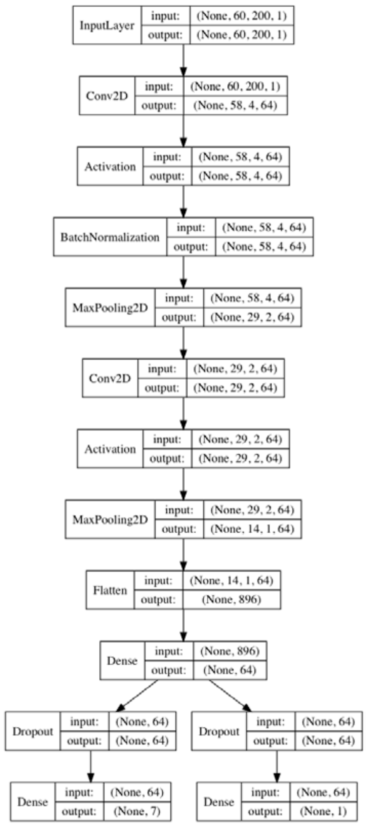
### 5.2 Reproducibility

In my development phase, I trained a bunch of CNN models. I choose the 30 best performing models as input to my final solution predictor. Clearly, regenerating these models, as happens with `train.sh`, won't produce the exact same models due to randomness. Since the `train.sh` pipeline only regenerates the specific CNN models that I choose during development, it's very well possible that the freshly trained models perform slightly worse than my final solution. If this appears to be the case, simply rerunning the training-pipeline will give different results.

## A CNN Layouts


InputLayer input: (None, 60, 250, 1) output: (None, 60, 250, 1)
Conv2D input: (None, 60, 250, 1) output: (None, 58, 4, 64)
Activation input: (None, 58, 4, 64) output: (None, 58, 4, 64)
BatchNormalization input: (None, 58, 4, 64) output: (None, 58, 4, 64)
MaxPooling2D input: (None, 58, 4, 64) output: (None, 29, 2, 64)
Conv2D input: (None, 29, 2, 64) output: (None, 29, 2, 64)
Activation input: (None, 29, 2, 64) output: (None, 29, 2, 64)
MaxPooling2D input: (None, 29, 2, 64) output: (None, 14, 1, 64)
Flatten input: (None, 14, 1, 64) output: (None, 896)
Dense input: (None, 896) output: (None, 64)
Dropout input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 7)

(a) Layout of the CNN. It takes an window of 60 time steps. In the first layer, the keV axis is reduced to size 4, which is more or less similar to taking inner products along the keV axis. It is trained as a classifier on 7 classes: no source, or any of the 6 source types.


InputLayer input: (None, 60, 200, 1) output: (None, 60, 200, 1)
Conv2D input: (None, 60, 200, 1) output: (None, 58, 4, 64)
Activation input: (None, 58, 4, 64) output: (None, 58, 4, 64)
BatchNormalization input: (None, 58, 4, 64) output: (None, 58, 4, 64)
MaxPooling2D input: (None, 58, 4, 64) output: (None, 29, 2, 64)
Conv2D input: (None, 29, 2, 64) output: (None, 29, 2, 64)
Activation input: (None, 29, 2, 64) output: (None, 29, 2, 64)
MaxPooling2D input: (None, 29, 2, 64) output: (None, 14, 1, 64)
Flatten input: (None, 14, 1, 64) output: (None, 896)
Dense input: (None, 896) output: (None, 64)
Dropout input: (None, 64) output: (None, 64)
Dropout input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 7)
Dense input: (None, 64) output: (None, 1)

(b) Similar layout to the left, but this model output consists of two parts: a classifier for any of the 7 classes, and a source location predictor. If there is a source in the window, then the location output is trained to return the source location inside the window.

## E 6th Place Writeup: cyril.v

# Detecting Radiological Threats in Urban Areas

topcoder

[redacted]
handle: cyril.v

## 1. Model summary

My solution is predominantly based on multiple deep learning models with convolutional and Long Short-Term Memory (LSTM) layers on binned energy events counts by time. Three deep learning models are trained with binned data sequences : source detect and type identifier to find probability of sequence containing artificial radiation source and select source type identifier, source window detector to find the probability of on already detected source sequence window to contain the least source distance and source timing model to find most probable time of the least source distance.

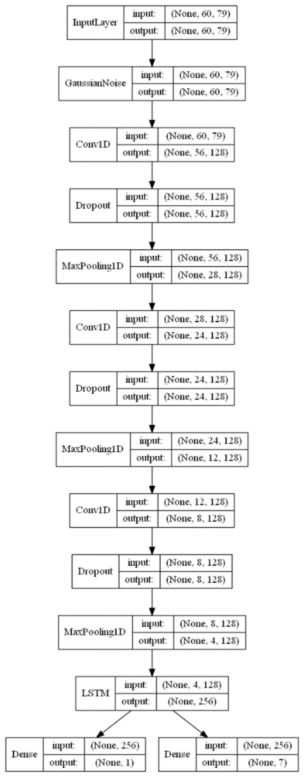
## 2. Model description

### a. Data preprocessing

Data are first binned the same way on each model with training and testing. For each run data are extracted from csv file (energy and last event timing). Cumulative sums are done on lasts events timing to obtain constant time offset of each event and convert to seconds. Events counts are binned with energy histogram on each time step selected of 0.5 second (with python numpy histogram2d function). Histogram bins are 79, spaced evenly on a log scale between 0 and 4000keV energy (python numpy geomspace function).

Each model gets in input window of 30 seconds of binned histogram, so 60 intervals histogram of 0.5 second resulting in input of shape 60x79. Training and validating windows data are selected specifically on each model. Testing windows data are from the sliding window of all histogram sequence.

## b. Source detect and type identifier


InputLayer | input: | (None, 60, 79)
output: | (None, 60, 79)
GaussianNoise | input: | (None, 60, 79)
output: | (None, 60, 79)
Conv1D | input: | (None, 60, 79)
output: | (None, 56, 128)
Dropout | input: | (None, 56, 128)
output: | (None, 56, 128)
MaxPooling1D | input: | (None, 56, 128)
output: | (None, 28, 128)
Conv1D | input: | (None, 28, 128)
output: | (None, 24, 128)
Dropout | input: | (None, 24, 128)
output: | (None, 24, 128)
MaxPooling1D | input: | (None, 24, 128)
output: | (None, 12, 128)
Conv1D | input: | (None, 12, 128)
output: | (None, 8, 128)
Dropout | input: | (None, 8, 128)
output: | (None, 8, 128)
MaxPooling1D | input: | (None, 8, 128)
output: | (None, 4, 128)
LSTM | input: | (None, 4, 128)
output: | (None, 256)
Dense | input: | (None, 256)
output: | (None, 1)
Dense | input: | (None, 256)
output: | (None, 7)

Source detect, and type identifier deep learning model is used to determinate if a sequence contains or not contains an artificial radiation source and to identify the source type.

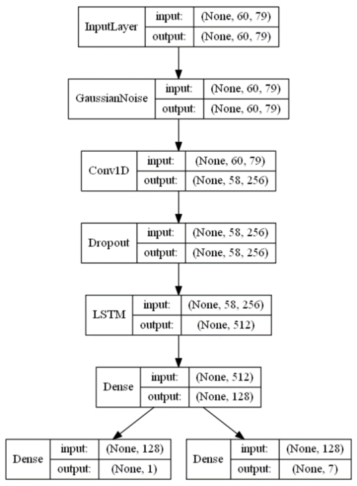
It is composed of three stacked modules of same layers in front of a LSTM layer with tanh activation function.

The stacked modules are composed of a convolutional layer with 128 filters and kernel size of 5, with dropout to avoid overfitting followed by a pooling layer.

The outputs layers are a dense layer with sigmoid activation function to obtain source detection and a dense layer with softmax activation function to get type identifier of source. Small random gaussian noise is applied to input data on training to avoid overfitting.

Training is done on 1 phase of 100 epoch and 9 phases of 20 epoch. On each phase 80% of data is randomly selected to train and 20% to evaluate. At each epoch best validation accuracy model is saved.

## c. Source window detector


InputLayer | input: | (None, 60, 79)
output: | (None, 60, 79)
GaussianNoise | input: | (None, 60, 79)
output: | (None, 60, 79)
Conv1D | input: | (None, 60, 79)
output: | (None, 58, 256)
Dropout | input: | (None, 58, 256)
output: | (None, 58, 256)
LSTM | input: | (None, 58, 256)
output: | (None, 512)
Dense | input: | (None, 512)
output: | (None, 128)
Dense | input: | (None, 128)
output: | (None, 1)
Dense | input: | (None, 128)
output: | (None, 7)

Source window detector deep learning model is used on sequence of data with source detected by the first model to evaluate the probability of slice window of a sequence to be in the least distance of radiation source.

It is composed of a convolutional layer with 256 filters and kernel size of 3, with dropout to avoid overfitting followed by a LSTM layer and a dense layer with sigmoid activation function.

The outputs layers are a dense layer with sigmoid activation function to obtain source detection and a dense layer with softmax activation function to get type identifier of source. Type identifier detected on this model is not used on testing but present to format model weights on training and obtain better accuracy. Small random gaussian noise is applied to input data on training to avoid overfitting.

Training is done with 2000 epoch. 75% of data is randomly selected to train and 25% to evaluate. Only sequence with artificial radiation source is used on training. At each epoch best validation loss model is saved.

## d. Source timing

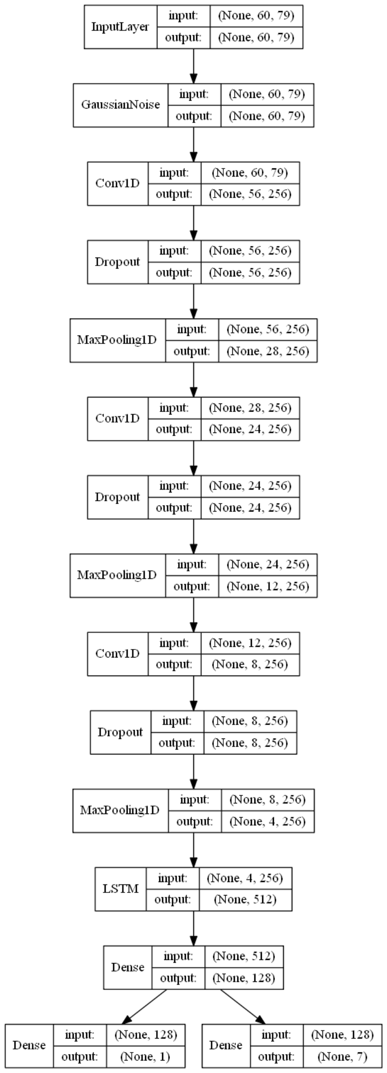
Source timing deep learning model is used to determinate time at which the detector was closest to the source in a sequence.

It is composed of three stacked modules of same layers in front of a LSTM layer with tanh activation function.

The stacked modules are composed of a convolutional layer with 128 filters and kernel size of 5, with dropout to avoid overfitting followed by a pooling layer.

The outputs layers are a dense layer with linear activation function to obtain source time in the window and a dense layer with softmax activation function to get type identifier of source. Type identifier detected on this model is not used on testing but present to format model weights on training and obtain better accuracy. Small random gaussian noise is applied to input data on training to avoid overfitting.

Training is done with 2000 epoch. 75% of data is randomly selected to train and 25% to evaluate. Sequence with no artificial radiation source is used on training but not in loss function of timing. At each epoch best validation accuracy model is saved.


InputLayer input: (None, 60, 79) output: (None, 60, 79)
GaussianNoise input: (None, 60, 79) output: (None, 60, 79)
Conv1D input: (None, 60, 79) output: (None, 56, 256)
Dropout input: (None, 56, 256) output: (None, 56, 256)
MaxPooling1D input: (None, 56, 256) output: (None, 28, 256)
Conv1D input: (None, 28, 256) output: (None, 24, 256)
Dropout input: (None, 24, 256) output: (None, 24, 256)
MaxPooling1D input: (None, 24, 256) output: (None, 12, 256)
Conv1D input: (None, 12, 256) output: (None, 8, 256)
Dropout input: (None, 8, 256) output: (None, 8, 256)
MaxPooling1D input: (None, 8, 256) output: (None, 4, 256)
LSTM input: (None, 4, 256) output: (None, 512)
Dense input: (None, 512) output: (None, 128)
Dense input: (None, 128) output: (None, 1)
Dense input: (None, 128) output: (None, 7)

### e. Solution export

On solution export, each data sequence are cuts with the sliding window of all histogram sequence. The three models are run on each window.
Presence of artificial source in sequence is detected with the first model if the maximum predict of windows of sequence is superior of 0.95 to avoid a maximum of false positive.
If presence of source is detected, 30 consecutive windows in sequence is selected with maximum sum of squared probability result obtained on source window detector model. Source time is obtained with the weighted average of the source timing model on the 30 windows selected. Weights are the power 4 of probability of source window detector.

## 3. Solution implementation

Models are implemented in python 3 with Keras and Tensorflow backend.
Training is done on training data with:

- main-hitid.py: Train source detect and type identifier deep learning model, export model in hitid31.best.hdf5 file
- main-in.py: Train source window detector deep learning model, export model in in.best9.hdf5
- main-time.py: Train source timing detector deep learning model, export model in hitidtime10.best.hdf5

Solution export is done on testing data with main-test.py

Python requirements are: tensorflow-gpu, keras, numpy, pandas and more_itertools

## 4. Solution testing

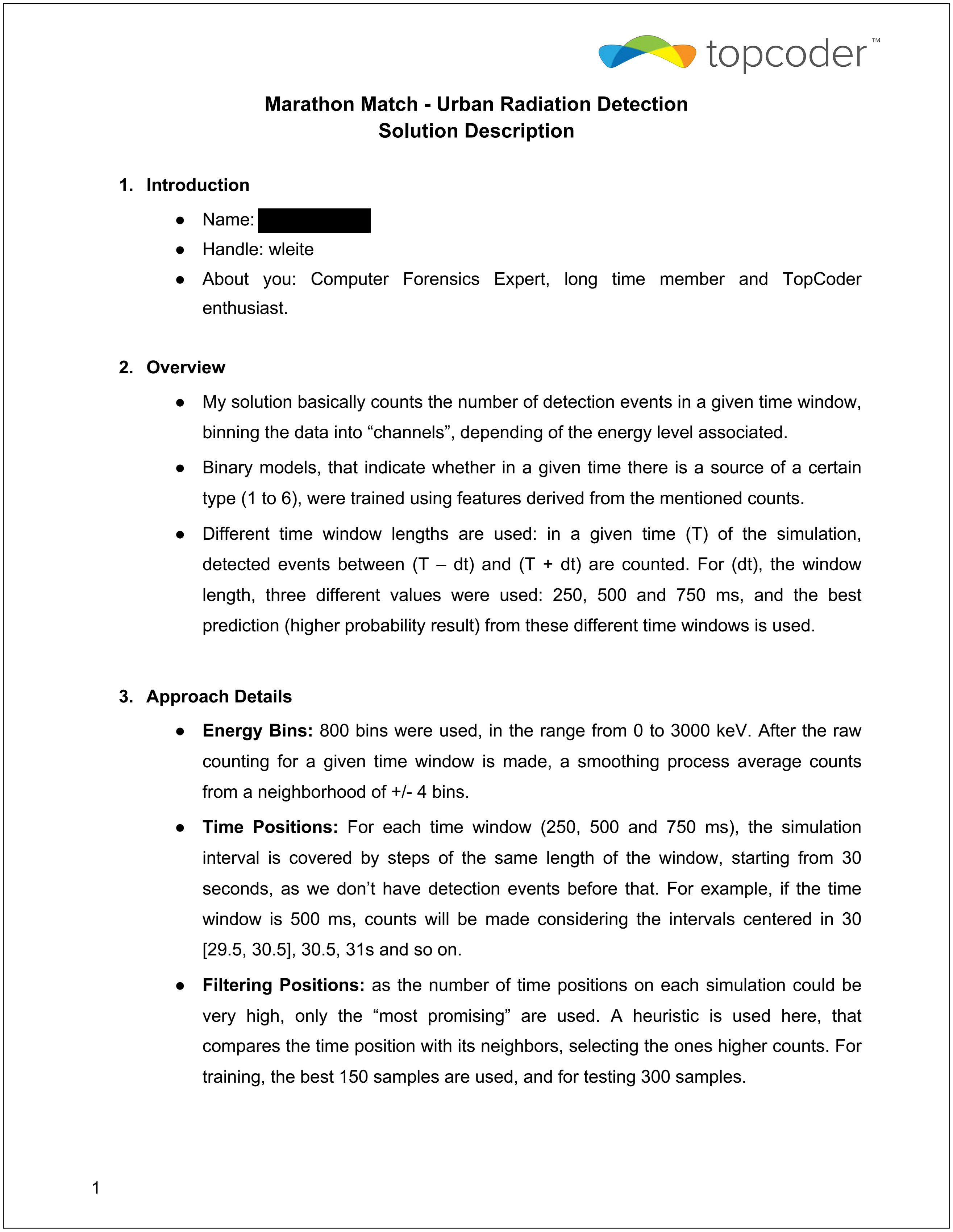
Solution container was tested with nvidia-docker on GPU enabled host p3.2xlarge AWS instance.

Provided zip file contains all models training and test source code with already trained submission models files.

Solution export with testing data finish in 30 minutes:
```
./test.sh /data/testing/ solution.csv
```

Training of models (output in current directory) finish in 3 hours and 15 minutes:
```
./train.sh /data/training/ /data/trainingAnswers.csv
```

## F 8th Place Writeup: wleite



topcoder™

**Marathon Match - Urban Radiation Detection**
**Solution Description**

1. **Introduction**
   - Name: [redacted]
   - Handle: wleite
   - About you: Computer Forensics Expert, long time member and TopCoder enthusiast.

2. **Overview**
   - My solution basically counts the number of detection events in a given time window, binning the data into "channels", depending of the energy level associated.
   - Binary models, that indicate whether in a given time there is a source of a certain type (1 to 6), were trained using features derived from the mentioned counts.
   - Different time window lengths are used: in a given time (T) of the simulation, detected events between (T – dt) and (T + dt) are counted. For (dt), the window length, three different values were used: 250, 500 and 750 ms, and the best prediction (higher probability result) from these different time windows is used.

3. **Approach Details**
   - **Energy Bins:** 800 bins were used, in the range from 0 to 3000 keV. After the raw counting for a given time window is made, a smoothing process average counts from a neighborhood of +/- 4 bins.
   - **Time Positions:** For each time window (250, 500 and 750 ms), the simulation interval is covered by steps of the same length of the window, starting from 30 seconds, as we don't have detection events before that. For example, if the time window is 500 ms, counts will be made considering the intervals centered in 30 [29.5, 30.5], 30.5, 31s and so on.
   - **Filtering Positions:** as the number of time positions on each simulation could be very high, only the "most promising" are used. A heuristic is used here, that compares the time position with its neighbors, selecting the ones higher counts. For training, the best 150 samples are used, and for testing 300 samples.

1

- **Model and Features:** the underlying model uses random forests of 1024 binary classification trees. There is one model for each type of source (1 to 6) and for each time window (250, 500 and 750). For the features, raw values of each "channel" (bin) are used, together with relative values, which compare counts for the current time position with adjacent ones, emphasizing variations on the observed values.
- **Hard Cases:** as a final attempt to improve the performance in "hard cases", which should be more frequent in testing set according to the problem statement, another set of models were trained using only the 1500 "harder" cases from the training set (instead the whole set). To identify these hard cases, the solution was executed twice using the "local test" mode, splitting training data into two equal parts (50/50). In the first run, the first half was used as training and the second was evaluated and compared with the ground truth. In the second execution, sets were inverted. All simulations that were incorrectly identified (wrong type of source or source far from the correct time) are included as "hard cases". The cases correctly identified but with a low confidence level (close to the used acceptance threshold of 0.152) are also added, until 1500 cases are selected. In total, 36 models were used: 2 (hard or regular) * 6 (source types) * 3 (250, 500 and 750 ms for time windows).
- **Inference:** positions in time, for each time window, are evaluated using the trained models and the best (higher confidence value is used). "Hard models" and the regular models are mixed using a weighted average, with 40% and 60% weights, respectively. A final minor optimization averages the value of predicted time of a source detection with the (up to) 5 higher confidence positions. That improved precision in local tests, but I am not sure how effective it was in the actual testing cases.

## 4. Dockerized Solution

It can be download from the following Google Drive link. It was tested using an AWS m4.10xlarge. It contains a README.txt file with useful information. Inference took about 1 hour to run in that AWS instance. Training took less than 24 hours in my home PC (24 threads), so it should faster than that in the used AWS instance. Training could be much faster with less tree (e.g. 128 instead of 1024 would reduce total time by a factor of 8), without losing much the quality of predictions.

[redacted]

## G 9th Place Writeup: smg478

03 May 2019

[redacted]

Topcoder handle: smg478

**Solution summary of Radiological Threat detection challange (9th place)**

In this challenge, I have developed an algorithm that is inspired by radiation detection theory and spectroscopy techniques. To built machine learning models, I have applied spectrum based features which considers important peak information as well. I have built a lightweight model that trains and predicts fast and is scalable to larger applications.

One of the interesting aspects of this competition was that the data was variant in both time and space dimensions. The signals (full-energy peaks) can be found from short time spans to long time spans, also from short distances to long distances. Thus, a preferable method to search for signals would be to look into smaller segments of data, rather than the full-run data. Therefore, my preferred method of building models was a sliding window approach in the full-run data.

1. Training data preparation:

Training data was generated by slicing the full-run data into smaller segments. Segment sizes were selected based on two scenarios: one was based on ***fixed*** number of counts (fixed sized windows), while another was based on ***variable*** number of counts, which was calculated from the *total number of counts* in a train file (variable size windows). Fixed size windows (e.g. window size = 3000 and 6000 counts) can serve as local time-invariant samples. However variable size windows (e.g. window size = *total counts* / 30) provides us an opportunity to look at the data from a global perspective. If a source was present, 7 different sized segments were generated from close to the source location, otherwise, less than 15 segments were **randomly** generated from uniformly distributed locations. Approximately 81,000 training segments of different sizes were created from 9700 competition data available. Fig 1 and 2 show schematic diagrams of train sample generation. Data from the first 30 seconds was also ignored during this process.

2. Feature generation:

First, a **gamma-ray spectrum** was generated from each segmented train sample. Selecting bins of 30 keV resulted in a 100 bin spectrum and values from each bin were treated as a feature. Next, another 51 features were calculated based on peak ratios. Bin counts of important peaks associated with a source were used to calculate **peak-to-peak ratios** and **peak-to-compton ratios**. These ratios play an important role in radiation detection and measurement and provide useful information to distinguish between different radioactive sources. In total, 151 features were used for machine learning model training.

3. Model construction:

Machine learning (ML) models experimented in this competition were mainly based on neural networks. I have designed a hybrid model of Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP) neural network that learns from 151 features. Though only CNN or MLP can be used as a model architecture, the experimental result shows combining both of them provides better accuracy and makes them less prone to overfit. In the hybrid model, 64 feature-maps were extracted from 151 input features using a 2-layer CNN model. The CNN features were then concatenated to original 151 features. Finally, a two-layer MLP network was used which takes input from all 215 (151+64) features before the classification layer. This model achieved a provisional score of **85.62** in the leaderboard (Table 1: expt#10). This model is referred to ANN-CNN model in table 1. Fig 4 shows the schematic diagram off **ANN-CNN** model architecture.

Further, I have experimented with Long Short Term Memory **(LSTM)** networks and Light Gradient Boosting Machine **(LGBM)** to compare the performance. All of the 3 models perform similarly in the validation data. However, my designed architecture was lightweight and fast to train and test.

4. Training:

Keras ML library was used for neural network construction and training. The ANN-CNN model was trained for 30 epochs with a learning rate of 0.001. Categorical cross entropy was used as a loss function. I have also experimented with focal loss. However, both loss functions performed similarly. While the ANN-CNN model takes about 10 minutes to train on 5-folds, LSTM and LGBM models take much higher time to converge (i.e. 25 minutes each) on an NVIDIA Titan X GPU.

5. Inference:

During inference, 200 equally spaced anchor points were selected from each test file. From each anchor point, the model predicted the probability of a source using 3 different window sizes. **Multiple window prediction** strategy can also be referred to Test Time Augmentation (TTA), and this improved the provisional leaderboard score considerably (Provisional LB: 82.3 to 85.3)(Table 1: expt # 4, 5, 6). However, doing more TTA would improve the score further, but for maintaining the time constraint, 3 TTA strategy was adopted. Predictions were ensembled from all 5 folds and finally, voting was used to decide source type and approximate location.

6. Finetune source location:

I have further adopted a rule-based method to finetune the source location. After finding the approximate location of source from neural network models, further peak search was carried out in the nearby area. The measure for the nearby area was selected based on the *total number of counts* in the test file (similar to the method of selecting a variable size window, except search area = *total counts* / 20, from approx. location). I have selected the location where the **highest number of counts for the associated peak** was found in a sliding window manner. This approach saved the time in finding the location by scanning a segment of the test file and

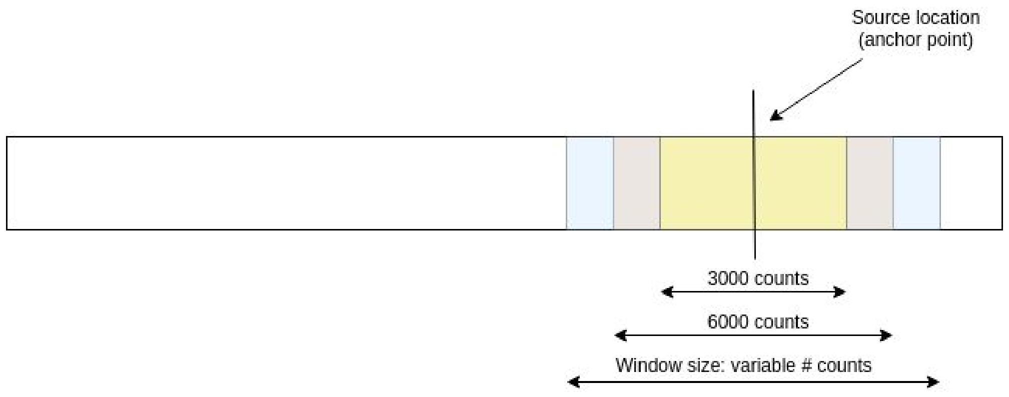


Fig 1. Training sample generation from runs with source

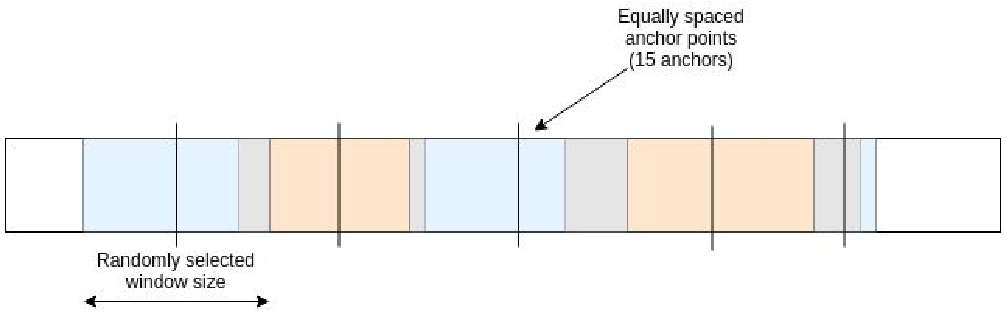


Fig 2. Training sample generation from runs without source

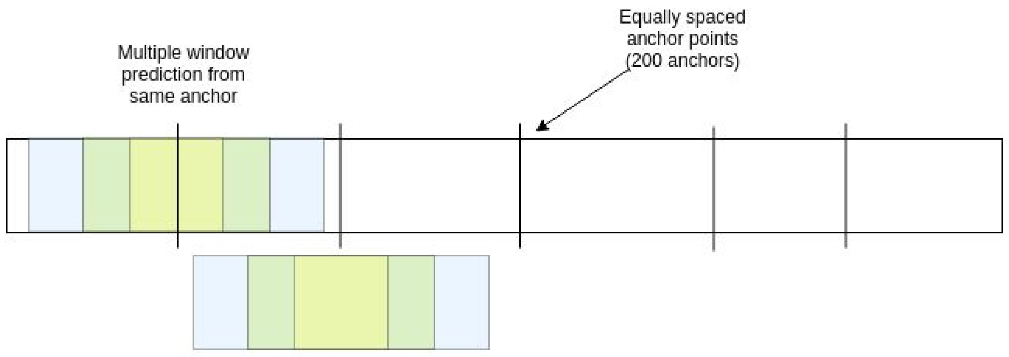


Fig 3. Prediction in a sliding window approach

produced a solid boost the provisional score from 80.0 to 82.3 in the leaderboard (Table 1: expt#3,4).

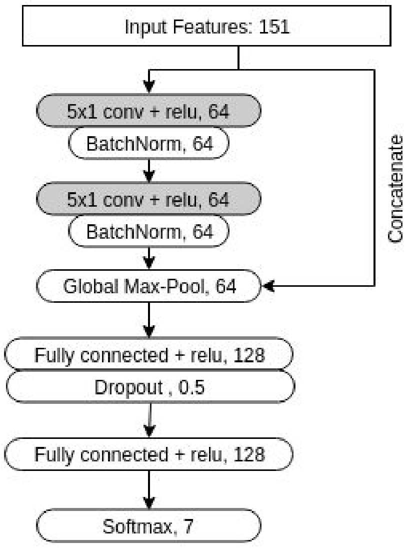


Fig 4. ANN-CNN neural network architecture

Table 1: Ablation study
(Prediction time was based on an 8-core Intel i7 processor with 32 GB RAM and SSD)

| Expt. No# | Description | Provisional score | Data processing time | Training time(GPU/CPU) | Prediction time (CPU) |
|---|---|---|---|---|---|
| 1 | **Single fold, CNN model**, full-length training on raw data **(100 features: 100 bin spectrum values only)**, **rule-based predictions on full-length data** | 67.0 | 0 | 2 min /30 min | 8 hr |
| 2 | Single fold, CNN, **segment-wise training** (100 features), **segment-wise predictions** | 74.0 | 30 min | 2 min / 30 min | 2 hr |
| 3 | Single fold, **ANN model**, segment-wise training (100 features), **normalize data**, segment-wise predictions | 80.0 | 30 min | 2 min / 30 min | 2 hr |
| 4 | Single fold, **ANN-CNN model** segment-wise training (100 features), normalize data, segment-wise predictions, **Rule-based time processing, no TTA** | 82.3 | 30 min | 2 min / 30 min | 3 hr |
| 5 | **3-fold** ANN-CNN models, **3-TTA** | 84.59 | 30 min | 6 min / 45min | 4 hr |
| 6 | **5-fold** ANN-CNN models, **5-TTA** | 85.29 | 30 min | 10 m/1.25 hr | 6 hr |
| 7 | 5-fold **ANN-CNN, LGB, LSTM** models, 5-TTA | 85.43 | 30 min | 1 hr / -- | 13 hr |
| 8 | 5-fold ANN-CNN, LGB, LSTM models, 5-TTA - **w/o time processing** | 84.09 | 30 min | 1 hr / -- | 12 hr |
| 9 | 5-fold ANN-CNN, LGB, LSTM models, **3TTA**, **peak-ratio features added** (151 features, 100 bin spectrum values + **51 hand engineered peak ratio features**) | 85.46 | 30 min | 1 hr / -- | 8 hr |
| 10 | 5-fold **ANN-CNN models**, 3TTA, peak-ratio features added (151 features, 100 bin spectrum values + 51 hand engineered peak ratio features) | 85.62 | 30 min | 10 min / 1.25 hr | 6 hr |
| 11 | 5 fold ANN-CNN, LGB, LSTM models, 3-TTA (151 features) - used **pseudo label** data for training from the test set (from 85.62 provisional score file) | 85.34 | 30 min | 30 min + 10 min + 6 hr +1 hr + 10 min = ~8 hr | 8 hr |

##############################################################################
**Code docomentation**
##############################################################################

The folder consists of 7 scripts that do data preparation, training and prediction. Scripts are written in python language.

Training and testing both can be done on a CPU based machine. However, training in a GPU is much faster. Testing using GPU doesn't concern much.

#=========================================================================
***# Script description***
#=========================================================================
**Data preparation**
01_make_slice_data.py
02_make_features.py
- Script 01 makes approximately 81,000 segmented data from 9,700 training data available and saves newly generated data on "wdata/training_slice" folder and corresponding answer file as 'wdata/trainingAnswers_slice.csv'.
- Script 02 takes the files generated by script 01, creates 151 features from each file and finally save everything as 'wdata/train_feature_bin_30_slice.csv'

**# Training**
03_train_ANN_CNN.py
- This script trains a hybrid model of convolutional neural network (CNN) and multi-layer perceptron (MLP) neural network using training features generated in script 02.
- Model weights will be saved in 'weights/' folder as well as in 'wdata/weights' folder

**# Inference**
06_predict_25.py
07_predict_3000.py
08_predict_6000.py
- 3 prediction files are identical except they predict on different segment (window) sizes. The prediction was carried out on 200 anchor points. These scripts use weights produced from script 03.
    - Script 06: window size = total counts in test file / 12
    - Script 07: window size = 3000 counts
    - Script 08: window size = 6000 counts
- 3 different thresholds (e.g. 3, 5 and 7 out of 200) were used to judge a test file as source positive or negative.
- The reason I have used 3 scripts for prediction instead of 1 because I found it little complicated to run parallel jobs with tensorflow models. So I have saved time by using 3 scripts running in parallel.

- Output files will be saved under ‘wdata/submits’ folder

**# Ensemble predictions**
09_vote_ensemble.py
- This script ensemble the prediction of the source type and location by voting style from the 9 output files produced from inference (script 06, 07, 08).
- The output file will be saved under ‘wdata/submits’ folder

**# Finetune source location**
10_timeProcess.py
- Outputs the final predictions and saves it to the current directory.

#=====================================================================
**How to run the code**
#=====================================================================
The code is expected to run in Docker container. Docker is assumed to be installed in the host computer. This code doesn’t  need a GPU for training or inference. Dockerfile file is sufficient for a cpu only machine. It installes necessary python dependancies on a Ubuntu 16.04 OS.

# start docker
*sudo service docker start*
# build solution from the folder that contains Dockerfile
*docker build -t smg478 .*
# Strat container
*docker run -v <local_data_path>:/data:ro -v <local_writable_area_path>:/wdata -it <id>*
# Inference on local built model
*bash test.sh /data/testing/ solution.csv*
# Ttrain
*bash train.sh /data/training/ /data/trainingAnswers.csv*
# Inference on newly trained model - produces solution file on current directory
*bash test.sh /data/testing/ solution.csv*

**Expected running time:**
Local PC config: Ubuntu 14.04, Intel i7 (8-core), 32 GB RAM, SSD
Disc space required: 7 GB for processed data file + 5 MB for model weights
*Training*
bash train.sh /data/training/ /data/trainingAnswers.csv
- (2.0 / 1.0) hr in a (CPU / GPU) based machine

*Testing*
bash test.sh /data/testing/ solution.csv
- 6 hr in a CPU based machine (+ 30 min, if processed data file needs to be generated again. Usually this file will be generated during the training phase (200MB))

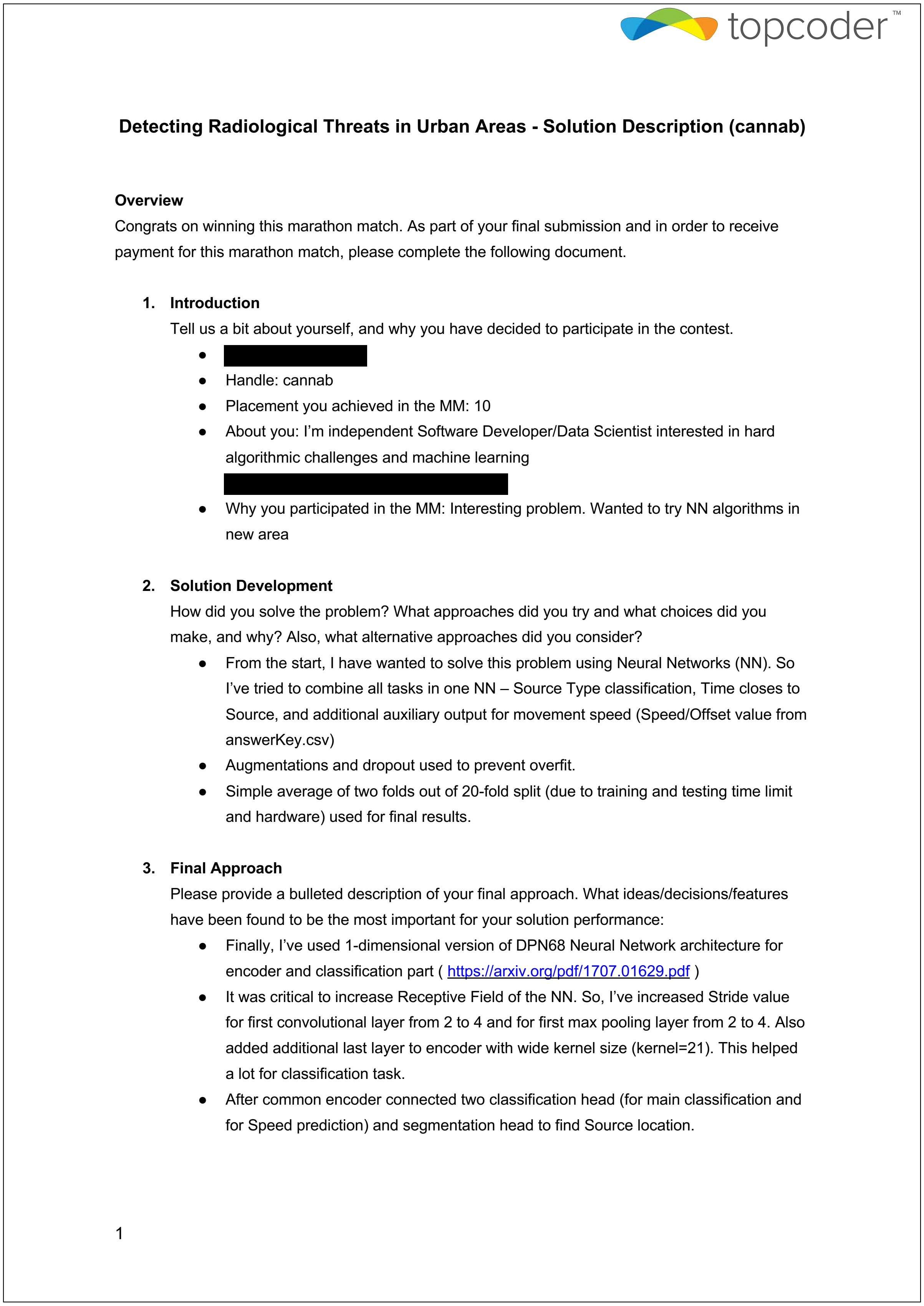
```
##########################################################################
# Important note on reproducibility
##########################################################################
```

The generation of final prediction uses some statistics from training data. Training statistics are calculated from the segmented data generated by script 01. Data segmentation process has a random variable for the runs that don't have any source (see Fig 2). Therefore, every time data processing is done, statistics will be a little different. If one uses the same processed train data for prediction, results will be the same every time.

However, I found a typo in my data processing script (script 01) in the later stage of this competition which produced wrongly labeled segmented data. And when I corrected the mistake, I accidentally overwrote the previous file that produced exactly my final submission (provisional 85.62). Therefore I am unable to produce the exact same submission now due to this missing statistics. Now in this folder, I have provided the correct version of script 01, and it will produce newly segmented train data as well as statistics. I expect by using the provided locally built model weight files, results will be close to 85.62+- 0.15.

Finally, retraining would produce similar results as well. I expect after retraining, the final/provisional score would be in a reasonable range.

## H 10th Place Writeup: cannab



topcoder™

**Detecting Radiological Threats in Urban Areas - Solution Description (cannab)**

**Overview**

Congrats on winning this marathon match. As part of your final submission and in order to receive payment for this marathon match, please complete the following document.

1. **Introduction**

   Tell us a bit about yourself, and why you have decided to participate in the contest.

   - [redacted]
   - Handle: cannab
   - Placement you achieved in the MM: 10
   - About you: I'm independent Software Developer/Data Scientist interested in hard algorithmic challenges and machine learning
     [redacted]
   - Why you participated in the MM: Interesting problem. Wanted to try NN algorithms in new area

2. **Solution Development**

   How did you solve the problem? What approaches did you try and what choices did you make, and why? Also, what alternative approaches did you consider?

   - From the start, I have wanted to solve this problem using Neural Networks (NN). So I've tried to combine all tasks in one NN – Source Type classification, Time closes to Source, and additional auxiliary output for movement speed (Speed/Offset value from answerKey.csv)
   - Augmentations and dropout used to prevent overfit.
   - Simple average of two folds out of 20-fold split (due to training and testing time limit and hardware) used for final results.

3. **Final Approach**

   Please provide a bulleted description of your final approach. What ideas/decisions/features have been found to be the most important for your solution performance:

   - Finally, I've used 1-dimensional version of DPN68 Neural Network architecture for encoder and classification part ( https://arxiv.org/pdf/1707.01629.pdf )
   - It was critical to increase Receptive Field of the NN. So, I've increased Stride value for first convolutional layer from 2 to 4 and for first max pooling layer from 2 to 4. Also added additional last layer to encoder with wide kernel size (kernel=21). This helped a lot for classification task.
   - After common encoder connected two classification head (for main classification and for Speed prediction) and segmentation head to find Source location.

1

- For segmentation part used simple 1-dimentional decoder architecture similar to Unet ( https://arxiv.org/pdf/1505.04597.pdf ). Ground truth area considered: abs(t - sourceTime) * speed_offset < 0.25
- For training as Neural Network inputs used crops with 49152 time points with 168 features vector for each timepoint:
  - Time from prev event, Energy Value, Energy Value / Time normalized with global std and mean values
  - Time from prev event, Energy Value, Energy Value / Time normalized with local (from current run) std and mean values
  - Energy counts in interval of +- 0.5s around current time point, clustered to 81 bins (with step of 50)
  - Clustered Energy value to 81 bins (with step of 50) for each time point (Just flag 1 or 0)
- Loss function used for training segmentation part is combination of Dice and Folca losses: (Dice + 10 * Focal). CrossEntropyLoss for classification and L1 loss for Speed.
- Simple augmentations used to prevent overfitting:
  - Skip random time point
  - Duplicate random time point
  - Random shift/scale values for some points
  - Test runs predicted using all run values. If Predicted Source Type > 0 and there are time points in segmentation output with probability > 0.5, Then center of largest predicted segment taken as Location of Source.

4. **Open Source Resources, Frameworks and Libraries**
   Please specify the name of the open source resource along with a URL to where it's housed and it's license type:
   - Anaconda as base Python 3 environment, www.anaconda.com
   - Pytorch, https://pytorch.org
   - Pretrained models, https://github.com/Cadene/pretrained-models.pytorch

5. **Potential Algorithm Improvements**
   Please specify any potential improvements that can be made to the algorithm:
   - Use for ensembling something like LightGBM to predict Source Type using many statistical features (planned but had no time for this)

6. **Algorithm Limitations**
   Please specify any potential limitations with the algorithm:
   Training and prediction are very slow.

7. **Deployment Guide**
   Please provide the exact steps required to build and deploy the code:

Dockerized version prepared as requested. For clean installation python 3 required with libraries (all in anaconda3 default installation): numpy, sklearn, + install Pytorch

8. **Final Verification**
   Please provide instructions that explain how to train the algorithm and have it execute against sample data:
   train.sh and test.sh scripts meet required specification.

9. **Feedback**
   Please provide feedback on the following - what worked, and what could have been done better or differently?
   - Problem Statement - ok. All clear
   - Data - ok. No leaks and other problems.
   - Contest – Very good contest. A lot of prizes, interesting task. Want more such contests)
   - Scoring - ok

**NOTE**: Please save a copy of this template in word format. Please do not submit a .pdf

# I 11th Place Writeup: pasda

# PASDA's Methodology: Detecting Radiological Threats in Urban Environments

[redacted]

*May 1, 2019*

## 1 Introduction

The challenge *Detecting Radiological Threats in Urban Areas* is about detecting, identifying, and locating radiological material using a moving sensor in an urban environment. See the website https://www.topcoder.com/challenges/30085346 for full details. The scenario is a sensor that moves in a straight line on a simulated road measuring gamma-ray energy levels (at apparently random times). If there is a radiological source present, it emits energy according its energy spectra. This is obfuscated by the presence of background radiation sources and reflections from the urban environment. For a set of testing runs, the objective is to determine: **detect** if there is a radiological source in the run, **identify** the source (1-6 sources each with shielded or non-shielded profiles), and **locate** the time at which the sensor was closest to the source.

The competition organizers have generated data from thousands of runs that mimic what would be acquired by a $2 \times 4 \times 16$ NaI(Tl) detector moving down a simplified street in a mid-sized U.S. city. Each run has been designed so that no source is located within the first 30 seconds of measurements, though the first 30 seconds could include events associated with gamma rays arising from an extraneous source located farther along the street.

## 2 Method Overview

### 2.1 Scan Statistic

Let `SourceID = k`, where $k = 0$ implies no source and $k \in \{1.0, 1.1, 2.0, 2.1, \ldots, 6.0, 6.1\}$ corresponds to one of the 6 radiological sources without shielding ($x.0$) and with shielding ($x.1$). Let $\tau \geq 30$ be the time (in secs) when the sensor is closest to the source. Note: the contest stated that the source would not be located within the first 30 seconds.

For each run, $\mathcal{H}_0$ is the null hypothesis of no source (i.e., $k = 0$) and $\mathcal{H}(k, \tau)$ (for $k \in K$ and $\tau \geq 30$) be the hypothesis that the source is $k$ and the sensor is closest to the source at time $\tau$.

Let $\Lambda(k, \tau)$ be the likelihood ratio:

$$\Lambda(k, \tau) = \frac{\Pr(Data \mid \mathcal{H}(k, \tau))}{\Pr(Data \mid \mathcal{H}_0)}$$

The *generalized likelihood ratio* (or *scan statistic*) plugs in the MLEs for $k$ and $\tau$

$$\Lambda^* = \frac{\max_{k,\tau} \Pr(Data \mid \mathcal{H}(k, \tau))}{\Pr(Data \mid \mathcal{H}_0)}$$

where the decision would be to choose $\mathcal{H}(\hat{k}, \hat{\tau})$, the MLE, when $\Lambda^*$ is suitably large. We can find the best threshold to be the one that maximizes performance over the training data (or hold out set).

#### 2.1.1 Likelihood Ratio

For a run with $n$ observations, the likelihood ratio (assuming independence between observations conditional on model parameters) is

1

$$\Lambda(k,\tau) = \frac{\Pr(Data \mid \mathcal{H}(k,\tau))}{\Pr(Data \mid \mathcal{H}_0)} \tag{1}$$

$$= \prod_{i=1}^{n} \frac{f(x_i|\mathcal{H}(k,\tau))}{f(x_i|\mathcal{H}_0)} \tag{2}$$

which is the product of density ratios at the observations $x_1, \ldots, x_n$. The null density, $f(x_i|\mathcal{H}_0)$, can be directly estimated from the training data (see below for details). However the density $f(x_i|\mathcal{H}(k,\tau))$ depends on how likely $x_i$ has come from source $k$ that is located at time $\tau$. That is, even if a source is nearby the sensor, it could still receive observations from the background.

Consider a single observation $x_i$ (the recorded energy at time $t_i$). If this observation actually came from Source $k$, then the (log) ratio

$$r_{ik} = \log \frac{f_k(x_i)}{f_0(x_i)},$$

where $f_k(x_i) = f(x_i \mid \mathcal{H}_k)$ is the density/pmf from source $k$ (estimated from the training data) and $f_0(x_i) = f(x_i \mid \mathcal{H}_0)$ is the density from the background, is expected to be greater than zero. However if it came from the background (e.g. $\mathcal{H}_0$), then $r_{ik}$ expected to be less than zero.

A formal way to deal with this is to model $f(x_i|\mathcal{H}(k,\tau))$ as a statistical mixture

$$f(x_i|\mathcal{H}(k,\tau)) = f_k(x_i)\pi_i(\tau) + f_0(x_i)(1-\pi_i(\tau))$$

where $\pi_i(\tau)$ is the probability that an observation at time $t_i$ would receive a measurement from a source location at time $\tau$. Applying this to the density ratios in (2) we get

$$\frac{f(x_i|\mathcal{H}(k,\tau))}{f(x_i|\mathcal{H}_0)} = \frac{f_k(x_i)\pi_i(\tau) + f_0(x_i)(1-\pi_i(\tau))}{f_0(x_i)}$$

$$= \frac{f_k(x_i)}{f_0(x_i)}\pi_i(\tau) + (1-\pi_i(\tau))$$

This could be estimated using, e.g., EM, in a mixture model formulation but with with some sort of prior on the $\pi_i(\tau)$ according to knowledge about how far a source will emit radiation. Since I had no idea about this and again (alas), limited time, I dropped this idea for now and focused on a computationally faster and simpler approach.

We used the statistic $R_{ik} = \max\{0, r_{ik}\}$ to measure the evidence that observation $i$ was from source $k$. The lower threshold at 0 limits the variability in the statistic and focuses attention on the observations that are more likely to be seen from the source. Building off this statistic, we estimated the likelihood ratio in (2) as

$$S(k,\tau,h) = \log\left(\hat{\Lambda}(k,\tau)\right)$$

$$= K(t_i - \tau; h)R_{ik}$$

which is in the form of a kernel regression where $K(t_i - \tau; h)$ is a Gaussian kernel with bandwidth (standard deviation) $h$. The idea is that if the source is location at time $\tau$, then it will produce the most observations at times close to $\tau$ so we weight the $R$'s according to how far they are away from $\tau$. In practice, we estimated this at a range of $\tau$'s spaced no more than 0.025 seconds apart. And choose the value that gives the largest score: $\tilde{S}(k,h) = \max_\tau S(k,\tau,h)$.

Considered bandwidths of $H = \{0.5, 0.75, 1, 1.25, 1.5\}$.

### 2.1.2 Back to the scan statistic

One the likelihood ratios are estimated the next step is to determine if $\mathcal{H}_0$ should be rejected and, if so, estimate $k$, $\tau$. One problem with using $\tilde{S}(k,h)$ is that its variance may differ over $k$ and $h$. Thus, if not accounted for the model may produce too many false alarms and over attribute to certain sources.

To help rectify this, we estimated the mean, $\mu_0(k,h)$, and standard deviation, $\sigma_0(k,h)$, of $\tilde{S}(k,h)$ when the data are generated from $\mathcal{H}_0$. Then we used the standardized score,

$$Z(k,h) = \frac{\tilde{S}(k,h) - \mu_0(k,h)}{\sigma_0(k,h)}$$

that will more correctly treat each $k$ and $h$ equally.

Finally, our scan statistic for a run is

$$T = \max_{k \in K, h \in H} Z(k,h)$$

and the estimates for source and event time are

$$(\hat{k}, \hat{\tau}) = \arg \max_{k \in K, h \in H} Z(k,h)$$

### 2.1.3 Threshold for hypothesis testing

We reject the null of no source if $T \geq \phi$, for some threshold $\phi$.

While the contest description stated that the distributions in the test data were the same as in the training data, it wasn't clear if that also applied to the distribution of sources and null runs. That is, do we expect 50% of runs to be null? Because this wasn't given I opted to set the threshold, $\phi$ by optimizing on the public scoreboard. Not ideal, but I didn't see there was an alternative without being given more info.

### 2.1.4 Thoughts

There are numerous ways to improve this model. It seemed like the sensor speed should be important. A Bayesian approach certainly has the potential to do better, especially in regard to the timing of the source (which has a clear shape). I only used the training runs data for the null runs; this had to be very limiting. There are numerous ways I can think to use the training runs that contain a source. And numerous other notes I have taken and hope to get time to pursue in the near future.

### 2.1.5 Remainder of the document

The remainder walks through the details of each step. All work was done in R, using the packages of `tidyverse`, `KernSmooth`, and `ks`.

## 3 Data Structure

We assume the following directory structure:

```
- data
  - training
    - 100001.csv
    - 100002.csv
    - ...
    - 109700.csv
  - testing
    - 200001.csv
    - ...
    - 215840.csv

- mydata
  - SourceData.csv
  - trainingAnswers.csv
- models.R
- score-testdata.R
```

This required unzipping `sourceInfo.zip` and extracting `SourceInfov3\SourceData.csv` as well as untaring and extracting all the runs files from `training.tar.gz` and `testing.tar.gz`.

## 4 Source Density

The first step in our process is to estimate the density of the energy in all 12 possible sources (6 source types and shielded/unshielded). We were provided the energy spectra for 5 of the 6 source types (the 6th had to be estimated). This turned out to be the binned counts (2 keV bin widths) in some sort of ideal condition (source is 1 meter away from the center of the detector in a vacuum).

### 4.1 Source Data

The Source Data is contained in the `sourceInfo.zip` file. Specifically, the file `SourceInfov3\SourceData.csv` is the actual data.

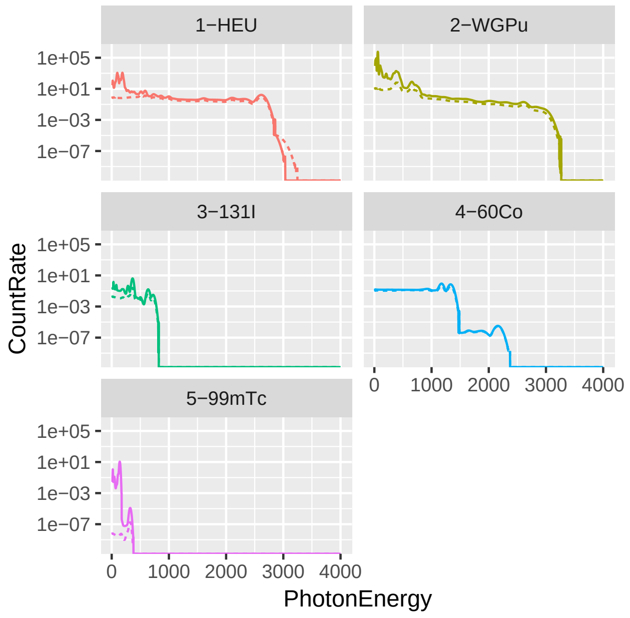
If a source is present in a run, it will be one of six types (plus a null):

| SourceID | Source Type |
|---|---|
| 0 | No Source |
| 1 | HEU: Highly enriched uranium |
| 2 | WGPu: Weapons grade plutonium |
| 3 | 131I: Iodine, a medical isotope |
| 4 | 60Co: Cobalt, an industrial isotope |
| 5 | 99mTc: Technetium, a medical isotope |
| 6 | A combination of 99mTc and HEU |

Energy spectra for each source type are shown below for a significant quantity of source types 1 and 2 and 1 microcurie ($\mu$Ci) for source types 3-5. In each figure, the solid curve shows the unshielded spectrum while the dashed curve shows the spectrum with 1 cm of lead shielding. The plots show sources 1 meter away from the center of the detector in a vacuum.

```r
Source = read_csv("data/SourceData.csv")

ggplot(Source, aes(PhotonEnergy, CountRate, color=factor(SourceID),
                   linetype=factor(Shielding))) +
  geom_line() +
  facet_wrap(~paste(SourceID, SourceType, sep="-"), nrow=3) +
  scale_y_log10(limits=c(10^(-10), 10^6)) +
  guides(color = FALSE, linetype=FALSE)
```



### 4.2 Estimating the Source Density

We actually treated the energy as discrete values and estimated the pmf using frequency histograms. The first step is to shift the value in PhotonEnergy back 1 to the bin midpoint.

```r
Source = read_csv("data/SourceData.csv") %>%
  mutate(PhotonEnergy = PhotonEnergy-1)
```

The next step is to make a function to linearly interpolate the intensity at 0.5 keV increments and rescale by summing over the total intensity to convert to a proper probability mass function.

```r
approx_df <- function(df, at=seq(11, 4001, by=.5)){
  tmp = approx(x=df$PhotonEnergy, y=df$CountRate, xout=at,
               rule=2)  # constant at end points
  tmp$y = pmax(0, tmp$y)    # ensure non-negative
  as_tibble(tmp) %>% transmute(energy=x, f=y/sum(y))
}
```

#### 4.2.1 Estimating Source 6

We were not provided the energy spectra for source 6, but told it was “a combination of 99mTc and HEU”. Without time to explore this any further, I just made it a 50-50 mix of the pmf from the two sources (Source 1 and 5).

```r
w = .50     # proportion from source 1

Source6 = Source %>%
  filter(SourceID %in% c(1, 5)) %>%
  ## Standardize
  group_by(SourceID, Shielding) %>%
  mutate(CountRate = CountRate/sum(CountRate)) %>% # standardize first
  group_by(Shielding, PhotonEnergy) %>%
  summarize(CountRate = CountRate[SourceID==1]*w + CountRate[SourceID==5]*(1-w)) %>%
  add_column(SourceType="HEU+99mTc", SourceID=6L, .before=1) %>%
  ungroup()

Source = bind_rows(Source, Source6)
```

### 4.2.2 Formatting the Density output

The density/pmf was then estimated for every source/shielding pair and converted to wide format:

```r
#-- Estimated (standardized) density at discrete energy levels for all sources
#   SourceID = 'SourceID.Shielding'
S = Source %>% group_by(SourceID, Shielding) %>% do(approx_df(.)) %>%
  ungroup() %>% unite(SourceID, SourceID, Shielding, sep=".") %>%
  spread(SourceID, f)
```

```r
head(S)
```

```
## # A tibble: 6 x 13
##   energy   `1.0`   `1.1`   `2.0`   `2.1`   `3.0`   `3.1`   `4.0`   `4.1`
##    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1   11   2.18e-4 4.02e-4 5.91e-4 6.77e-4 6.57e-4 6.92e-4 3.54e-4 4.01e-4
## 2   11.5 2.18e-4 4.02e-4 5.91e-4 6.77e-4 6.57e-4 6.92e-4 3.54e-4 4.01e-4
## 3   12   2.18e-4 4.02e-4 5.91e-4 6.77e-4 6.57e-4 6.92e-4 3.54e-4 4.01e-4
## 4   12.5 2.92e-4 3.68e-4 8.01e-4 6.21e-4 6.03e-4 6.35e-4 3.25e-4 3.68e-4
## 5   13   3.67e-4 3.33e-4 1.01e-3 5.65e-4 5.48e-4 5.77e-4 2.96e-4 3.35e-4
## 6   13.5 4.41e-4 2.99e-4 1.22e-3 5.09e-4 4.94e-4 5.20e-4 2.67e-4 3.02e-4
## # ... with 4 more variables: `5.0` <dbl>, `5.1` <dbl>, `6.0` <dbl>,
## #   `6.1` <dbl>
```

### 4.2.3 Thoughts

I didn't have time to explore the impact of intensity/rate. So I converted it all to a density and hoped that the overall rate varies more by the run factors (distance in road, buildings, etc.) that ignoring the rate won't harm too much in the prediction of source.

## 5 Null (No Source) Density

In order to use my intended test statistic, I also needed to estimate the null density, or the density of the background radiation when there was no source present.

First, I had to load the runs data that corresponded to no source. This required accessing the `trainingAnswers.csv` file which contains the solutions to the training runs.

```r
runs = read_csv("data/trainingAnswers.csv", col_types="ccd")
```

These are the First 6 rows of `trainingAnswers.csv`:

```r
head(runs) %>% knitr::kable()
```

| RunID | SourceID | SourceTime |
|---|---|---|
| 100001 | 0 | 0 |
| 100002 | 0 | 0 |
| 100003 | 0 | 0 |

| RunID | SourceID | SourceTime |
|---|---|---|
| 100004 | 0 | 0 |
| 100005 | 0 | 0 |
| 100006 | 0 | 0 |

The `RunID`'s with `SourceID = 0` correspond to the runs with no source.

The next step was to extract a random 100 runs.

```
#-- Set data dir
data.dir = "data/training"

#-- Sample 100 null runs
set.seed(2019)
RunID = runs %>% filter(SourceID == 0) %>% pull(RunID) %>%
  sample(size=100) %>% sort()


Y = tibble()
tt = LAM = numeric(length(RunID))
for(id in RunID){
  f = paste0(file.path(data.dir, id), ".csv")
  X = read_csv(f, col_names=FALSE, col_types="nn") %>%
    transmute(energy=X2) %>% count(energy) %>%
    add_column(RunID=id)
  Y = bind_rows(Y, X)
}
```

Due again to time constraints, I just aggregated the energy values over all times in all runs and didn't think about the uncertainty in this.

```
Y2 = Y %>% group_by(energy) %>% summarize(n = sum(n))
```

To help smooth over sparse regions, I then used kernel density estimation to make estimates at every 0.5 energy units increments (same as for the known Sources) and then converted that estimate to a pmf in the same way that we did for the other Sources.

```
m0 = ks::kde(Y2$energy, h=1, w=Y2$n,
             eval.points=seq(11, 4001, by=.5),
             positive=FALSE) %>%
             {tibble(PhotonEnergy=.$eval.points, density=.$estimate)}


#-- Baseline (standardized) density
B = approx_df(rename(m0, CountRate=density))
```

This created a density/pmf estimate from the background sources.

```
head(B) %>% knitr::kable()
```

| energy | f |
|---|---|
| 11.0 | 0.0001295 |
| 11.5 | 0.0001417 |
| 12.0 | 0.0001539 |
| 12.5 | 0.0001662 |

8

| energy | f |
|---|---|
| 13.0 | 0.0001784 |
| 13.5 | 0.0001906 |

### 5.0.1 Thoughts

There was so much unexplored here! I initially expected that the runs where along the same street and no we could use position/location information to model the uncertainty in the background radiation. Reading the contest description seemed to imply that all runs were on different roads so there was no way to connect a run in the training data to a run in the test data. If time permitted, I would have tried to match the first 30 seconds of test runs to some training runs to try to identify the road. There was also an issue about sensor speed, which I also didn't have time to explore.

The variability in the background radiation was just ignored. But I did do some initial plotting of the background density across runs and found some substantial variation. Unfortunately, I didn't have the time to explore how to incorporate this.

## 6 Test Statistics

The value $R_{ik} = \max\{0, r_{ik}\}$ can be calculated in advance to enable a quick look up at run time

```
#-- Get density Ratios

dens_ratio <- function(f.k, f, eps=1e-20){
  logr = log(f.k + eps) - log(f + eps)  # add small eps so f is not too close to 0
  pmax(0, logr)                         # assume negative values imply non-source
}

#-- Matrix of log density ratio
R = left_join(S, B, by="energy") %>%
  mutate_at(vars(-energy, -f), function(f.k) dens_ratio(f.k, .$f)) %>%
  select(-f)

#-- score_run()
#----------------------------------------------------------------------#
# Assigns a log density ratio score to every observation
#
# Because we only evaluate energy levels at discrete values, the energy
# is rounded to the nearest integer and constrained to be within the
# range of 11 through 4001.
# Then the log density ratio score, for every source, is retrieved for
#  the energy levels in the runs.
#----------------------------------------------------------------------#
score_run <- function(X, R){
  X %>% mutate(energy = round(energy/.5)*.5,  # round to 1/2
               energy = pmax(energy, 11),   # ensure at least 11
               energy = pmin(energy, 4001)) %>%  # ensure at most 4001
    left_join(R, by="energy")
}
```

The value $S(k, \tau, h)$ is the smoothed values of $R$ over time. The following function calculates this:

```
#-- smooth()
#----------------------------------------------------------------------#
```

```r
# Run kernel regression on the log density ratios
#
# Main idea is to find the time point that has the largest log density
# score, which indicates the sensor is close to a specific source.
# Inputs:
#   x: vector of times
#   Y: matrix of log density scores. One column per source.
#   bw: bandwidth for kernel smoothing. I think its sd of normal kernel,
#       this wasn't initially apparent from the locpoly() function
#   ngrids: number of grid points at which to return estimates. This is
#       only set to reduce time for computation. The default is to calculate
#       every .05 seconds.
# Outputs:
#   matrix with:
#     - first column the times at which estimate were made
#     - other columns for each SourceID
# Notes:
#   - requires KernSmooth function locpoly()
#   - to better identify the time the source is closest, we may want
#     to up ngrids?
#   - only returns values in [30, max(T)-4] seconds.
#     Since no event <30 and edge effects can impact this implementation
#     restrict source to be located within 4 seconds of end of run.
#--------------------------------------------------------------------#

smooth <- function(x, Y, bw=2, ngrids=NULL){
  xrng = c(25, max(x))  # no sources within first 30 seconds
  if(is.null(ngrids)) ngrids = 1+min(max(diff(xrng)/.025, 2500),6000) %>% ceiling()
  YHAT = matrix(0, ngrids, ncol(Y)); colnames(YHAT) = colnames(Y)
  for(j in 1:ncol(Y)){
    y = Y[,j,drop=TRUE]
    lp = KernSmooth::locpoly(x, y, bandwidth=bw, gridsize=ngrids,
                             degree=1,
                             range.x=xrng, truncate=FALSE)
    YHAT[,j] = lp$y
  }
  ok = (lp$x >= 30.0 & lp$x <= (max(x)-4) )
  cbind(x=lp$x[ok], YHAT[ok,])
}
```

## 6.1 Standardization

The value $Z(\tau, h)$ is obtained by standardizing $\tilde{S}(\tau, h)$, in a $Z$-score fashion, by subtracting the mean and standard deviation estimated from null (no source) runs. We used a sample of 900 runs to estimate.

```r
#--------------------------------------------------------------------#
#-- Estimate mu_0 and sigma_0
#   Estimate the mean and standard deviation of the test statistics
#   under some random runs from the null model
#--------------------------------------------------------------------#


#-- Set directory of training "runs" data
data.dir = "data/training"
```

```r
#-- Sample runs
n.sample = 900
set.seed(2019)
RunID = runs %>% filter(SourceID == 0) %>% pull(RunID) %>%
  sample(size=n.sample) %>% sort()


#-- Set bandwidths
BW = c(.5, .75, 1, 1.25, 1.5)

#-- Evaluate Runs
OUT = tibble()

pb = progress_estimated(n.sample)
for(id in RunID){
  #-- Read in runs data
  f = paste0(file.path(data.dir, id), ".csv")
  tau = filter(runs, RunID == !!id) %>% pull(SourceTime)
  X = read_csv(f, col_names=FALSE, col_types="nn") %>%
    transmute(time = cumsum(X1)/10^6, energy=X2)

  #-- Score runs
  A = score_run(X, R)
  for(j in 1:length(BW)){
    sm = smooth(A$time, select(A, -time, -energy), bw=BW[j])
    tau.hat = sm[apply(sm[,-1], 2, which.max),1]
    score = apply(sm[,-1], 2, max)
    out = tibble(runid=id, bw=BW[j], SourceID=colnames(sm)[-1], true.source=0,
                 tau.hat, dtau = tau - tau.hat,
                 score)
    OUT = bind_rows(OUT, out)
  }
  pb$tick()$print()
}


#-- Estimate distribution of score/Z under the null of no source
Z = OUT %>% group_by(SourceID, bw) %>%
  summarize(mu = mean(score), sd=sd(score))
```

```r
head(Z) %>% knitr::kable()
```

| SourceID | bw | mu | sd |
|---|---|---|---|
| 1.0 | 0.50 | 0.2943204 | 0.0114164 |
| 1.0 | 0.75 | 0.2829799 | 0.0095228 |
| 1.0 | 1.00 | 0.2766328 | 0.0084592 |
| 1.0 | 1.25 | 0.2724958 | 0.0078184 |
| 1.0 | 1.50 | 0.2695023 | 0.0072691 |
| 1.1 | 0.50 | 0.2718233 | 0.0125629 |

## 6.2 Scoring the Test Runs

The test data is in the file `testing.tar.gz` and contains the time and energy for 15840 runs.

```r
#-- Settings
data.dir = "data/testing"

ANS = read_csv("data/submittedAnswers.csv")


#-- Set bandwidths
BW = c(0.5, 0.75, 1, 1.25, 1.5)

#-- Evaluate Runs
OUT = tibble()

pb = progress_estimated(nrow(ANS))
for(i in 1:nrow(ANS)){
  #-- Read in runs data
  runid = ANS$RunID[i]
  f = paste0(file.path(data.dir, runid), ".csv")
  X = read_csv(f, col_names=FALSE, col_types="nn") %>%
    transmute(time = cumsum(X1)/10^6, energy=X2)

  #-- Score runs
  A = score_run(X, R)

  out.j = tibble()
  for(j in 1:length(BW)){
    sm = smooth(A$time, select(A, -time, -energy), bw=BW[j])
    tau.hat = sm[apply(sm[,-1], 2, which.max),1]
    score = apply(sm[,-1], 2, max)
    out = tibble(runid, bw=BW[j],     # runid=id
                 SourceID=colnames(sm)[-1],
                 tau.hat,
                 score)
    out.j = bind_rows(out.j, out)
  }
  out = out.j %>% left_join(Z, by=c("bw", "SourceID")) %>%
    mutate(Z = (score - mu)/sd) %>%
    group_by(SourceID) %>%
    ## Use bw with *max* Z
    summarize(
      runid = runid[1],
      tau.hat = tau.hat[which.max(Z)],
      bw = bw[which.max(Z)],
      Z = max(Z),
      score = score[which.max(Z)]) %>%
    filter(Z == max(Z)) %>%
    select(runid, bw, SourceID, tau.hat, score, Z)
    OUT = bind_rows(OUT, out)
  pb$tick()$print()
}
```

Finally, a threshold is selected and all test runs are assigned a source and time (if source is not 0).

```r
#-- Make submission data

thres = 2.21

solution = OUT %>% mutate(source = Z >= thres) %>%
  mutate(runid = ANS$RunID) %>%   # made mistake in above code
  mutate(SourceID = stringr::str_sub(SourceID, 1, 1),
         SourceID = ifelse(source, SourceID, 0L),
         SourceTime = ifelse(source, tau.hat, 0.00)) %>%
  select(RunID = runid, SourceID, SourceTime)
```

## J 12th Place Writeup: ZFTurbo

# Detecting Radiological Threats in Urban Areas

Author: ZFTurbo [redacted]
Provisional Score: 78.64456
Final score: 68.63696

### Requirements

Python 3.5 and GPU based system with nvidia-docker installed.

### Solution overview

Solution consists of 4 neural net models. Final solution file is the ensemble of predictions of these 4 models.

**Model 1**: It's segmentation model with UNET based decoder and ResNet50 as backbone, which is constructed using Convolution1D (instead of known version of ResNet50 for image classification task, where Convolution2D is used). It also has additional classification block which predicts class (one of 0, 1, 2, 3, 4, 5, 6).
Input size: (32768, 3)
Output 1 size: 32768
Output 2 size: 7

**Model 2**: Classification model with ResNet50 backbone which constructed using Convolution1D. It has 2 outputs first predict location of source (if any), second predicts type of source.
Input 1 size: (32768, 3)
Input 2 size: 16
Output 1 size: 128
Output 2 size: 7

**Model 3**: Classification model with DenseNet121 backbone which constructed using Convolution1D. It has 2 outputs first predict location of source (if any), second predicts type of source.
Input 1 size: (16384, 3)
Input 2 size: 16
Output 1 size: 128
Output 2 size: 7

**Model 4**: Recurrent neural net model based on bidirectional GRU layers with Attention block. It has 2 outputs first predict location of source (if any), second predicts type of source.
Input 1 size: (512, 192)
Input 2 size: 16
Output 1 size: 128

Output 2 size: 7

- All models trained on raw data provided in CSV files without rescaling or normalization.
- There are 3 stacked waveforms given for model 1-3: timing, energy and normed energy (energy / timing). For last RNN model it also reshaped to (512, 64*3).
- Only part of original waveform is used by each model. Third model takes 16384 consecutive points, others take 32768 consecutive points.

**Note 1**: 2nd input for models 2-4 contains some normalized statistics about overall waveform. Statistics includes: length, overall timing, minimum energy, maximum energy, mean energy, std energy, median energy, max timing, mean timing, std timing, median timing, normalized energy min, normalized energy max, normalized energy median, normalized energy mean, normalized energy std.
Logic behind this is following: we give to model only part of waveform for analysis, so statistics about full waveform can be useful for model to have some info about scaling, mean signal on full waveform, etc.

**Note 2**: My latest submit was ensemble of 4 models, but probably only single model (I think it's **Model 3**) can give comparable or even better result comparing to ensemble.

## Training

Training of models performed on GPU with usage of Keras module over tensorflow backend with Adam optimizer. I used infine batch generator.

| **Model** | **Batch size** | **Learning rate** | **Epochs** | **Steps per epochs** | **Patience** |
|---|---|---|---|---|---|
| Model 1 (Unet + ResNet50) | 48 | 0.0005 | 200 | 300 | 50 |
| Model 2 (ResNet50) | 120 | 0.0005 | 500 | 300 | 100 |
| Model 3 (Densenet121) | 96 | 0.0005 | 250 | 300 | 60 |
| Model 4 (GRU + Attention) | 240 | 0.0005 | 250 | 300 | 60 |

Table 1. Training parameters for models

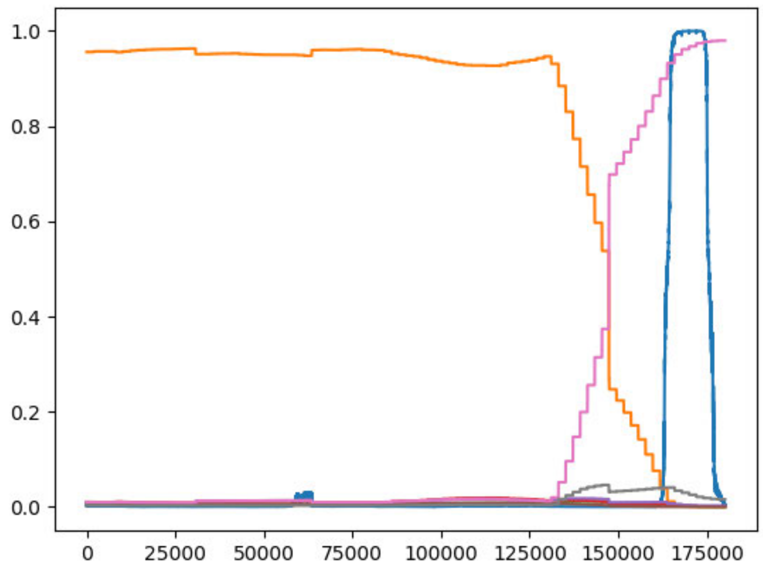
Batch size is mostly limited by available GPU memory. Each batch consists of following 3 types of cases:

1) 50% of batch is zero parts from zero source class
2) 25% only parts where source is far from current location
3) 25% only parts where source is located on extracted waveform

## Inference

Inference is made with sliding window approach. Initial waveform is cut on set of smaller segments with some step. Step is lower than size of waveform to have some overlapping segments. Typical step value equal to (input_size // 64). Then predicted waveforms concatenated and averaged in overlapped parts. Lower step size gives better accuracy but requires more time for prediction.

Example of prediction from single model and single run file are shown on figure (Test: 200006):



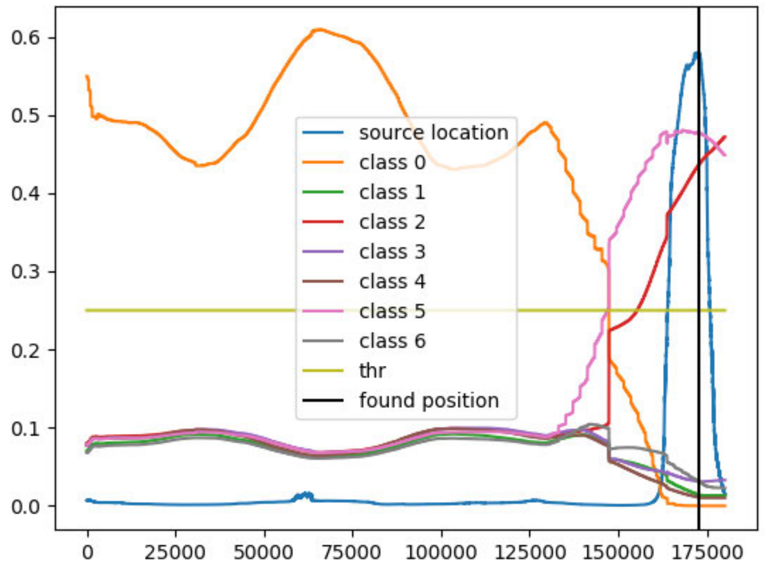
Here: blue - probability of close source, orange - probability of class 0, other colors - probability of classes 1-6.

All models have unified format of predictions.

## Ensemble

Ensemble is made directly on waveforms using simple average from 4 models. Optimal THR=0.25 was found on validation and used on test set. If probability of source is higher than THR at some point then we predict non-zero class.

Typical prediction from ensemble of 4 models and single run file (Test: 200006) is shown on figure:


source location
class 0
class 1
class 2
class 3
class 4
class 5
class 6
thr
found position